

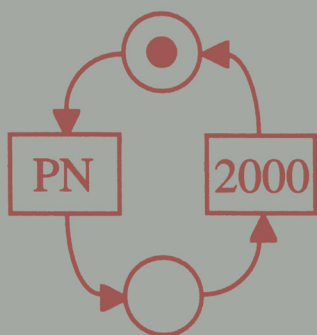
Lecture Notes in Computer Science

1825

Mogens Nielsen Dan Simpson (Eds.)

Application and Theory of Petri Nets 2000

21st International Conference, ICATPN 2000
Aarhus, Denmark, June 2000
Proceedings



Springer

Lecture Notes in Computer Science

Edited by G. Goos, J. Hartmanis and J. van Leeuwen

1825

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Singapore

Tokyo

Mogens Nielsen Dan Simpson (Eds.)

Application and Theory of Petri Nets 2000

21st International Conference, ICATPN 2000
Aarhus, Denmark, June 26-30, 2000
Proceedings



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Mogens Nielsen
University of Aarhus, Department of Computer Science
Ny Munkegade bldg 540, 8000 Aarhus C, Denmark
E-mail: mn@brics.dk

Dan Simpson
University of Brighton, Watts Building
Lewes Road, Brighton BN2 4GJ, East Sussex, UK
E-mail: Dan.Simpson@brighton.ac.uk

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Application and theory of Petri nets 2000 : 21st international
conference ; proceedings / ICATPN 2000, Aarhus, Denmark, June 26 - 30,
2000. Mogens Nielsen ; Dan Simpson (ed.). - Berlin ; Heidelberg ; New
York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ;
Tokyo : Springer, 2000
(Lecture notes in computer science ; Vol. 1825)
ISBN 3-540-67693-7

CR Subject Classification (1998): F.1-3, C.1-2, G.2.2, D.4, D.2, J.4

ISSN 0302-9743

ISBN 3-540-67693-7 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag is a company in the BertelsmannSpringer publishing group.
© Springer-Verlag Berlin Heidelberg 2000
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Boller Mediendesign
Printed on acid-free paper SPIN: 10721145 06/3142 5 4 3 2 1 0

Preface

Organising Committee

Tools Demonstration

Programme Committee

Referees

..

..

.. ..

.. ..

..

Table of Contents

Invited Papers

	
	
	
m	m	9

Full Papers

m	m	m	...
k	m	
		
m	..	m	..
		
..	m	
		
	m	
		

	m	
	
	
	“ ”	
	m
m		...
	m	m
	
m	
	m	m
	
m		
	m
	m
m	 9
	m	mm
		..

Tools Presentations

	m	
	
	

.....		
fl	fl
Author Index	

Hardware and Petri Nets: Application to Asynchronous Circuit Design

1 ky² y³
4 k 5
1 i i oli i l o o
o o l lo i
jordic@lsi.upc.es
2 l o o io ill o o 9 9
mkishine@ichips.intel.com
3 o i i i l 9
alex.kondratyev@theseus.com
4 i i i i i ll i i l
lavagno@uniud.it
5 i i o l o o o i i
l o o l o o l o
Alex.Yakovlev@newcastle.ac.uk

Abstract. o o i i i i i li i i o
o o i li o i i
o i o i o i i i
io l o l o i io i o i l i
o l i io o o o i i i i

1 Introduction

m m m
2 2 m m y m
m m m
m z y y m k y m
m m m
m m clock m
y m y y y
y k m y y
y z m m
y k m y y
k k initiation and completion of
several actions m y
y m k y
0 y

1 ll l i o l o o o i io o
li il i STG

o i o ll l

STG m m y y z y m m

y petrify y y

k m y y m y

3 Implementability Properties

y m y *speed-independent*

z y

y m y 2

m m

3.1 State Graph

SG y y STG y m m k m

STG SG y

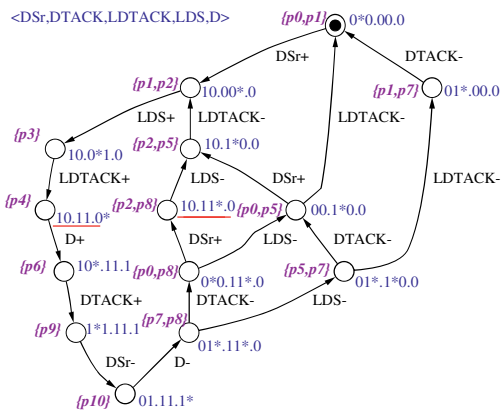


Fig. 3. y

SG k y

k k y

m k m k y

m k p₇, p₈ 0 . .0 DSr D 0

i

li

io

o

o

o

i

i

i

DTACK

LDTACK

LDS

DTACK

LDS

y

0

m

k

p_0, p_1

3.2 Properties for Implementability

m

m

– Boundedness

STG

SG

– Consistency

STG

– Completeness of state encoding

m

– Persistency

y

k

hazards

SG

y

y

m

k

p_4

p_2, p_8

m

p_2, p_8

LDS–

p_4

D

m

m

y

y

m

y

4 Logic Synthesis

y

z

y

m

y

m

m

y

m

y

–

SG

y

m

y

–

m

y

next-state

–

4.1 Complete State Coding

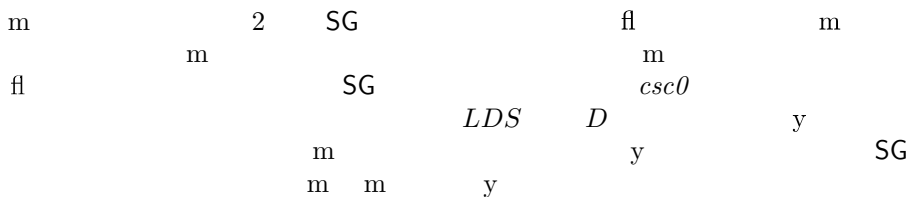
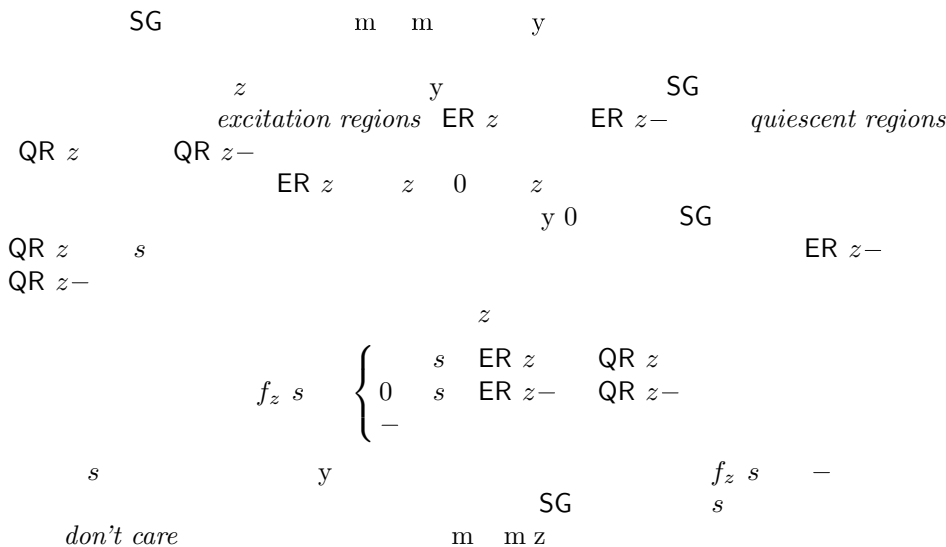


Fig. 4. SG

4.2 Next-State Functions



i li io o o o i i i
m m z
m m
m k
m y y SG
m
D LDTACK csc0 LDS D csc0
DTACK D csc0 DSr csc0 LDTACK
k y y y m
m m y m
m z 22 y m z m
y m
m m
— m m
— y
— y y y
y
m 2 y
c a b c a b c
a b k
m m m m
y m y
2 20

4.3 Size of the State Space

m
y m y z
y m y m
y y m
y m *Burst-mode automata*
y y k m
m y
m m
y m y m y
y m k m
m y y m m
m m

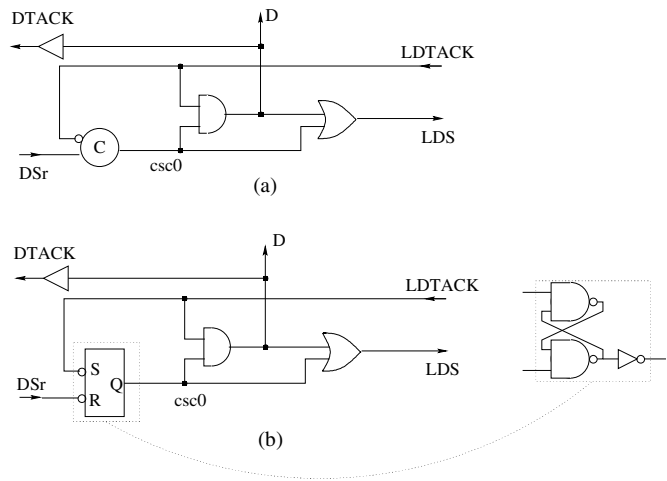
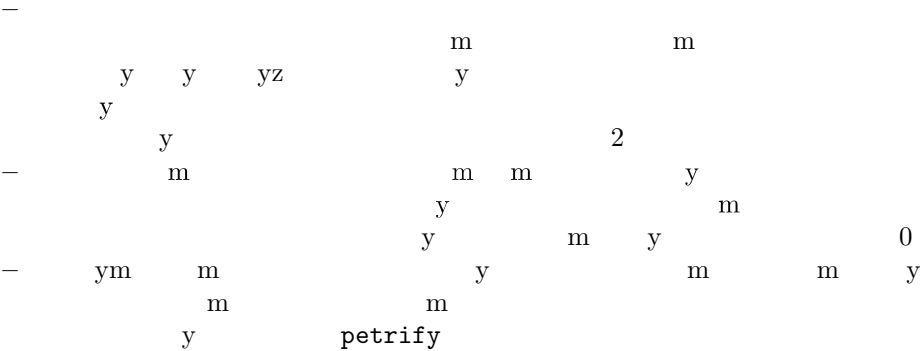
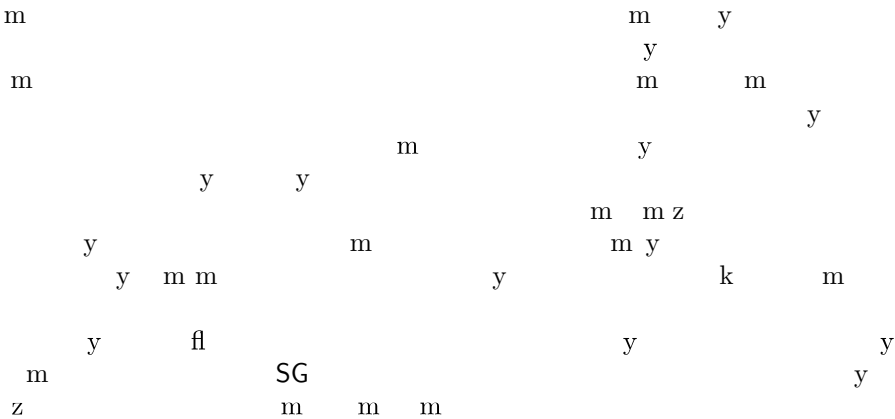


Fig. 5. m m 2



5 Back-Annotation



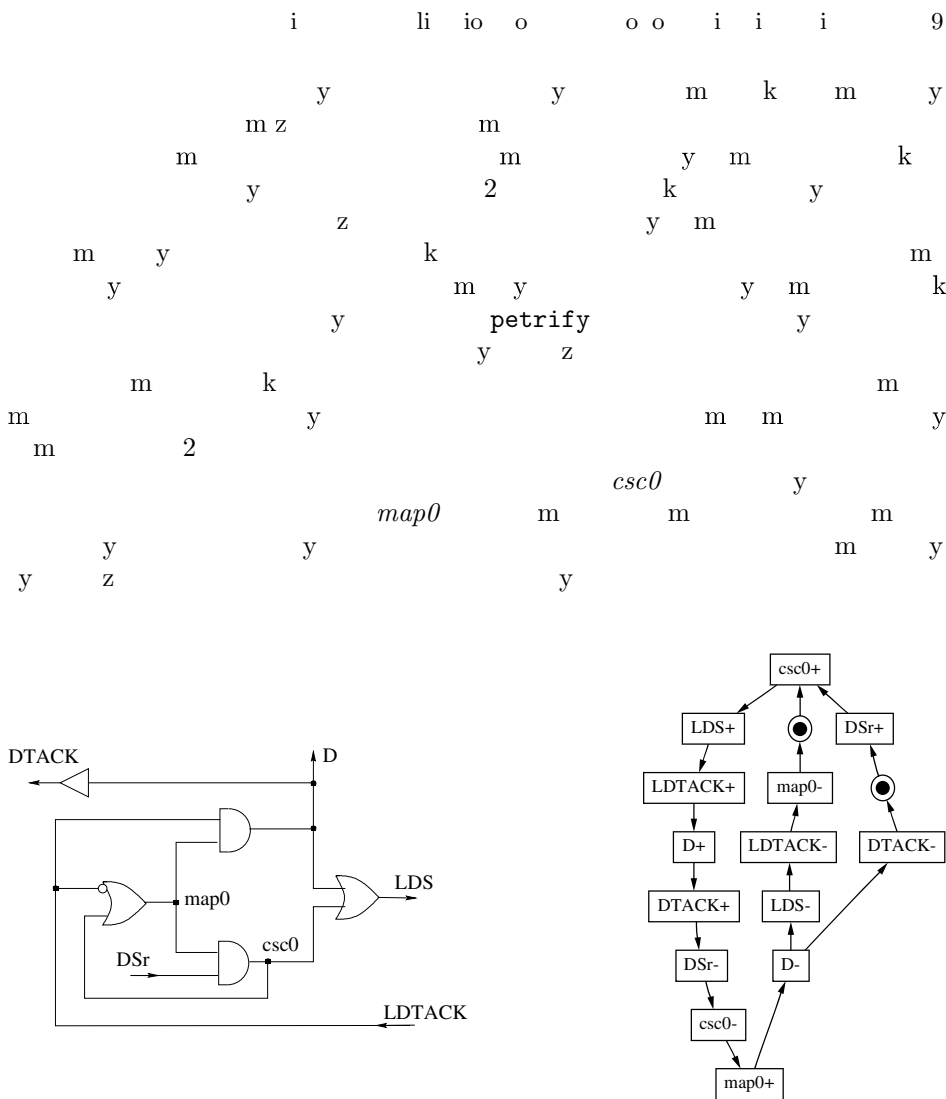


Fig. 6. 2

6 Analysis and Formal Verification

- *Property verification.* y k m
- m y

o i o ll l

m m y k

- m y y 2

– *Implementation verification.* y m k

y m m

m m m
- *Performance analysis and separation between events*

m y

m z m m 0

6.1 Techniques

- m

m j k y y y m m

m y
- ym y m m y m m

y m
- m m

m y
- m y 2

y m y
- 2 y k m

acyclic

m k y m m

y y fl y

m y

m

m y m y STG m 2

m

m m k m m 2

y y m k y y y m 2

m y m k y y y y

i li io o o o i i i

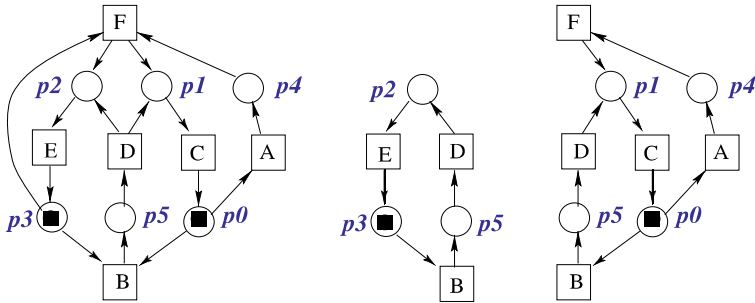


Fig. 7. STG

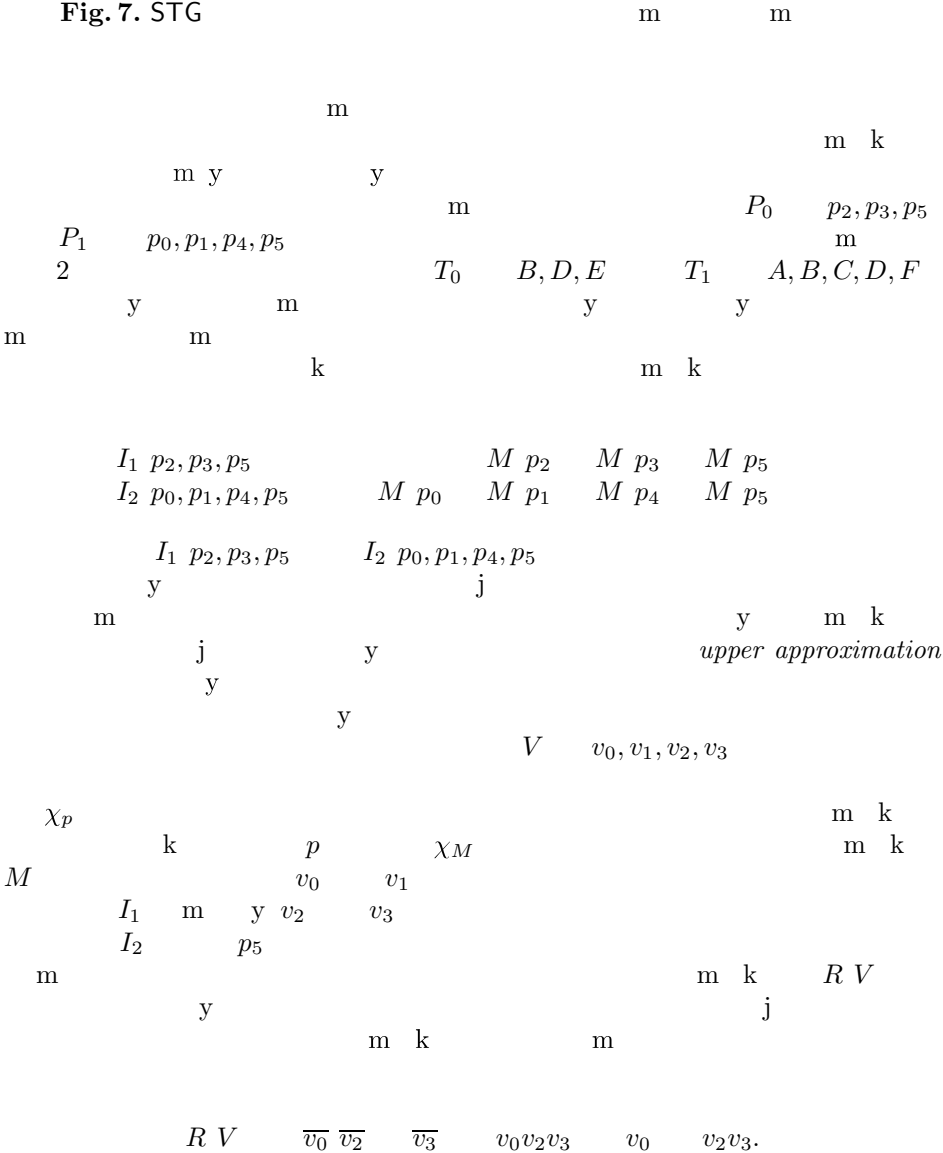


Table 1.

m k

l	$v_0\ v_1\ v_2\ v_3$	χ_p
p_2		$\overline{v_0\ v_1}$
p_3		$\overline{v_0}v_1$
p_5		$v_0v_2v_3$
p_0		$\overline{v_2\ v_3}$
p_1		$\overline{v_2}v_3$
p_4		$v_2\overline{v_3}$

i	χ_M
$\{p_0, p_3\}$	$\overline{v_0\ v_1\ v_2\ v_3}$
$\{p_5\}$	$v_0v_2v_3$
$\{p_1, p_2\}$	$\overline{v_0\ v_1\ v_2}v_3$
$\{p_1, p_3\}$	$\overline{v_0\ v_1\ v_2}v_3$
$\{p_0, p_2\}$	$\overline{v_0\ v_1\ v_2\ v_3}$
$\{p_3, p_4\}$	$\overline{v_0}v_1v_2\overline{v_3}$
$\{p_2, p_4\}$	$\overline{v_0\ v_1}v_2\overline{v_3}$

m $R\ V$ y D F 2 m

y D F

m k

D F

D F χ_{p_5} χ_{p_3} χ_{p_4}

m k

$R\ V$ D F .

y D F

m k

y D F y

y m 2

7 Conclusions

m y m y

m y m m

— y m

— y m

— m

— m m k

² i o il i i l i o o l o l
o ill i o o l i io i oli o i
i

o i o ll l

l ii i ol o i *Acta Informatica*
9 99

i ill *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits* i i i i io 9 9

o il *Acta Informatica* 99

o oi i ilo oi o o i i io o
l io *Proc. International Workshop on Computer Aided Verification* 99 i i i i

o i l o i 99
o *Communicating Sequential Processes* i ll 9
l o o illo l o i o

o o i io o i o *IEEE Transactions on Computers* o 99

i l i i l o l i i o
Concurrent Hardware: The Theory and Practice of Self-Timed Design

i i ll l o i o il o 99

9 l o i l i i l i i l *Formal Methods in System Design* 99

l o i l i i l ol i
io o i i *IEEE Transactions on Computer-Aided Design* 9 9 99

o l o ol o il l o i o o o l io o
l ol o o io *Hardware Design and Petri Nets* l i li

i o o l o io i i lli *Algorithms for Synthesis and Testing of Asynchronous Circuits* l i li 99

ill *Symbolic Model Checking* l i li 99

ill o i i io o o o i i i
ol i *Proc. International Workshop on Computer Aided Verification* 99

l o o i i ili i *Bell System Technical J.* 9 9

oo i o il i *Automata Studies*
9 9

i ll o o lo i li io o i o io o i
Symposium on the Application of Switching Theory to Space Technology
9 9 o i i 9

i ll o o o o i i *Proceedings of an International Symposium on the Theory of Switching*
i i il 9 9

9 i o i l i li io *Proceedings of the IEEE* il 9 9

i *IEEE Transactions on VLSI Systems* i o i o o i i 9 99

o i o o i o i o o o
i *Proc. International Conf. Computer Design (ICCD)*
o o i 99

o o ll l o o i o
oli l i o i *Application and Theory of Petri Nets 1999*
o i o i 999

i li io o o o i i i

 i o o i o ll l o iol oi l
 o o i o i i *IEEE Transactions on*
Computer-Aided Design 9 o 99
 i *Kommunikation mit Automaten* i o i "
 ll i 9 i l o i
 petrify ool o i o i o o o oll
 li o i i
 i i o i o *Lectures on Petri Nets I: Basic Models*
 ol 9 o *Lecture Notes in Computer Science* i l 99
 iol oi o i o ll i o i io o o o i
 i o l i o i *16th International Confer-*
ence on the Application and Theory of Petri Nets ol o *Lecture Notes*
in Computer Science 9 99
 o l ol i l o l i o i
 o *Proceedings of International Workshop on Timed Petri Nets* 99
 o i o l l 9 o o i
 9 l i o o io *Lecture Notes*
in Computer Science, Advances in Petri Nets 1990 ol 9
 i l 99

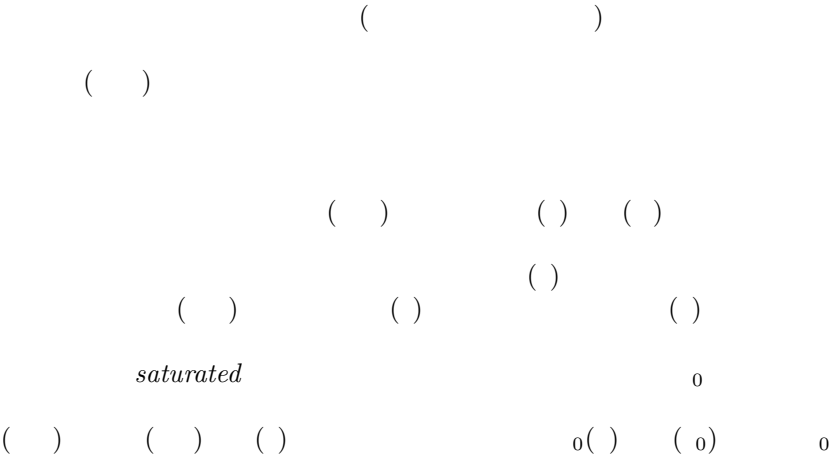
Region Based Synthesis of P/T-Nets and Its Potential Applications

IRISA, campus de Beaulieu, F35042 Rennes Cedex
darondeau@irisa.fr

Abstract. This talk is an informal presentation of ideas put forward by Badouel, Bernardinello, Caillaud and me for solving various types of P/T-net synthesis problems, with hints at the potential role of net synthesis in distributed software and distributed control. The ideas are theirs as much as mine. The lead is to start from Ehrenfeucht and Rozenberg's axiomatic characterization of behaviours of elementary nets, based on regions, to adapt the characterization to P/T-nets in line with Mukund's extended regions with integer values, and to profit from algebraic properties of graphs and languages for converting decision problems about regions to linear algebra.

1 Nets and Regions

$$\begin{array}{c}
 \mathcal{E} \qquad \mathcal{E} \\
 \\
 \mathcal{E} \qquad \qquad \qquad \left(\begin{array}{cc} & \\ & \end{array} \right) \quad \left(\begin{array}{cc} 1 & 2 \\ & \end{array} \right) \\
 1 \qquad \qquad \qquad 2 \\
 \\
 \mathcal{E} \qquad \qquad \left(\begin{array}{cc} & \\ & \end{array} \right) \qquad \mathcal{E} \left(\begin{array}{cc} & \\ & \end{array} \right) \qquad \mathcal{E} \quad \mathcal{E} \\
 \qquad \qquad \left(\begin{array}{cc} & \mathcal{E} \end{array} \right) \qquad \mathcal{E} \\
 \\
 \mathcal{E} \\
 \\
 \left(\begin{array}{cc} & \end{array} \right) \quad \mathcal{E} \\
 \left(\begin{array}{cc} & \end{array} \right) \qquad M \stackrel{-e}{\longrightarrow} M' \\
 M(p) \stackrel{N(p,e)}{\longrightarrow} M'(p) \\
 \\
 \qquad \qquad \qquad \left(\begin{array}{cc} & \mathcal{E} \end{array} \right) \\
 \mathcal{E} \qquad m \stackrel{(w_1,w_2)}{\longrightarrow} m' \qquad 1 \qquad - \quad 1 + \quad 2
 \end{array}$$



SYNET

2
A Method for Synthesizing P/T-Nets from Finite Automata, with Potential Applications to Distribution

$$1 \quad (\quad n \quad o) \quad + \quad 0 \quad (\quad)$$

$$\begin{aligned} & \left(\begin{array}{c} 0 \\ \vdots \\ 1 \end{array} \right) \quad \left(\begin{array}{c} 1 \\ \vdots \\ 1 \end{array} \right) \quad \left(\begin{array}{c} n \\ \vdots \\ n \end{array} \right) \\ 0 & \left(\begin{array}{c} \vdots \\ 0 \end{array} \right) \quad \left(\begin{array}{c} i \\ \vdots \\ i \end{array} \right) \quad \left(\begin{array}{c} i \\ \vdots \\ i \end{array} \right) \quad \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \\ & \left(\begin{array}{c} i \\ \vdots \\ i \end{array} \right) \quad \left(\begin{array}{c} i \\ \vdots \\ i \end{array} \right) \quad \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \quad \left(\begin{array}{c} 0 \\ \vdots \\ 0 \end{array} \right) \quad 0 \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \\ & \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \quad \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \quad \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \quad \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \quad \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \quad \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \\ & \sum_{i=1}^n \gamma_t \left(\begin{array}{c} i \\ \vdots \\ i \end{array} \right) \quad \left(\begin{array}{c} i \\ \vdots \\ i \end{array} \right) - \left(\begin{array}{c} i \\ \vdots \\ i \end{array} \right) \\ \gamma_t & \quad \gamma_t \left(\begin{array}{c} i \\ \vdots \\ i \end{array} \right) \quad \gamma_t \quad i \\ & 0 \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) + \sum_{i=1}^n s \left(\begin{array}{c} i \\ \vdots \\ i \end{array} \right) \quad \left(\begin{array}{c} i \\ \vdots \\ i \end{array} \right) - \left(\begin{array}{c} i \\ \vdots \\ i \end{array} \right) - \\ s & \left(\begin{array}{c} i \\ \vdots \\ i \end{array} \right) \quad s \quad i \quad s \\ & \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \quad 0 \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \quad i \quad i \quad \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \\ & \mathcal{C} \quad \sum_{j=1}^m j \quad j \quad j \quad \mathcal{I}_+ \\ & \quad j \quad j \quad \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \\ & \mathcal{C} \quad \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \quad \mathcal{C} \\ & \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \quad \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \quad j \\ & \mathcal{C} \quad 1 \quad m \\ & \left(\begin{array}{c} \vdots \\ \vdots \end{array} \right) \end{aligned}$$

$$0(\quad) + \sum_{i=1}^n (\quad i) \quad (\quad i - \quad i) - \sum_{i=1}^n (\quad i) \quad (\quad i - \quad i)$$

$$\left(\begin{array}{c} 1 \\ m \end{array} \right)$$

$$1 \qquad m$$

deterministic

(\ominus)

$$\Psi \left(\begin{matrix} W \\ 1 \end{matrix} \right) = \left(\begin{matrix} \ominus \\ m \end{matrix} \right) \quad W$$

$$0(\quad) + \sum_{i=1}^n (\quad_i) - (\quad_i - \quad_i) - \sum_{i=1}^n (\quad_i) - (\quad_i - \quad_i)$$

()

()

$$P$$

K

K

$$\begin{pmatrix} P & K \\ P & K \end{pmatrix} \begin{pmatrix} P & K \\ P & K \end{pmatrix}$$

References

1. Chernikova, N.V.: Algorithm for finding a general formula for the non-negative solutions of a system of linear inequalities. U.S.S.R. Computational Math. and Math. Physics **5** no.2 (1965) 228–233
2. Arnold, A., Nivat, M.: Comportements de processus. Les mathématiques de l'informatique, colloque AFCET (1982) 35–68
3. Ehrenfeucht, A., Rozenberg, G.: Partial (Set) 2-Structures; *part I*: Basic Notions and the Representation Problem. Acta Informatica **27** (1990) 315–342
4. Ehrenfeucht, A., Rozenberg, G.: Partial (Set) 2-Structures; *part II*: State Spaces of Concurrent Systems. Acta Informatica **27** (1990) 343–368
5. Mukund, M.: Petri Nets and Step Transition Systems. Int. Journal of Found. of Comp. Science **3** no.4 (1992) 443–478
6. Rudie, K., Wonham, W.M.: Think Globally, Act Locally: Decentralized Supervisory Control. IEEE Trans. on Automatic Control **37** no.11 (1992) 1692–1707
7. Droste, M., Shortt, R.M.: Petri Nets and Automata with Concurrency Relations - an Adjunction. Semantics of Programming Languages and Model Theory, M. Droste and Y. Gurevich eds (1993) 69–87
8. Badouel, E., Bernardinello, L., Darondeau, Ph.: Polynomial algorithms for the synthesis of bounded nets. Proc. CAAP, LNCS **915** (1995) 364–378
9. Jancar, P., Moeller, F.: Checking Regular Properties of Petri Nets, Proc. Concur, LNCS **962** (1995) 348–362
10. Badouel, E., Darondeau, Ph.: Dualities between nets and automata induced by schizophrenic objects. Proc. CTCS, LNCS **953** (1995) 24–43
11. Desel, J., Reisig, W.: The Synthesis Problem of Petri Nets. Acta Informatica **33** (1996) 297–315
12. Badouel, E., Darondeau, Ph.: Theory of regions. Lectures on Petri Nets I: Basic Models, LNCS **1491** (1998) 529–586
13. Bernardinello, L.: Propriétés algébriques et combinatoires des régions dans les graphes et leur application à la synthèse de réseaux. Thèse, Univ. Rennes (1998)
14. Darondeau, Ph.: Deriving Unbounded Petri Nets from Formal Languages. Proc. Concur, LNCS **1466** (1998) 533–548
15. Badouel, E.: Automates réversibles et réseaux de Petri, dualité et représentation : le problème de synthèse. Document d'habilitation **31**, Irisa (1999)
available from <http://www.irisa.fr/EXTERNE/bibli/habilitations.html>
16. Caillaud, B.: Bounded Petri-net Synthesis Techniques and their Applications to the Distribution of Reactive Automata. JESA **9–10** no.33 (to appear)
17. Darondeau, Ph.: On the Petri Net Realization of Context-Free Graphs. (to appear)
18. Synet: <http://www.irisa.fr/pampa/LOGICIELS/synet/synet.html>

n n n

engels|reiko|sauer@upb.de

[illegible]

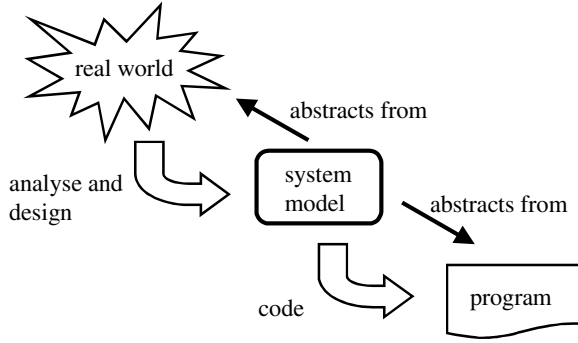


Figure 1.

documentation

ward engineering

reverse engineering

structured methods

abstract data type

The diagram is a complex web of symbols and text. At the top left, the word "paradigm" is written in a cursive font. At the top right, the phrase "object-oriented" is written in a similar cursive font. In the center, the phrase "universal language" is written in a cursive font. At the bottom, the word "profiles" is written in a cursive font. The diagram is filled with numerous small, lowercase letters, including 'n', 'm', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', and 'a'. These letters are arranged in a way that suggests a flow or relationship between the main concepts. There are also some mathematical symbols, such as a minus sign (-) and a plus sign (+). The overall layout is dense and intricate, with many lines and curves connecting the various elements.

2 Language Overview

n n n n n n n
 n n n n n n n
 n n 99 n n n n
 n n n n n n n
 n n n n n n n
 n n n n n n n
 n n n n n n n
 n n n n n n n

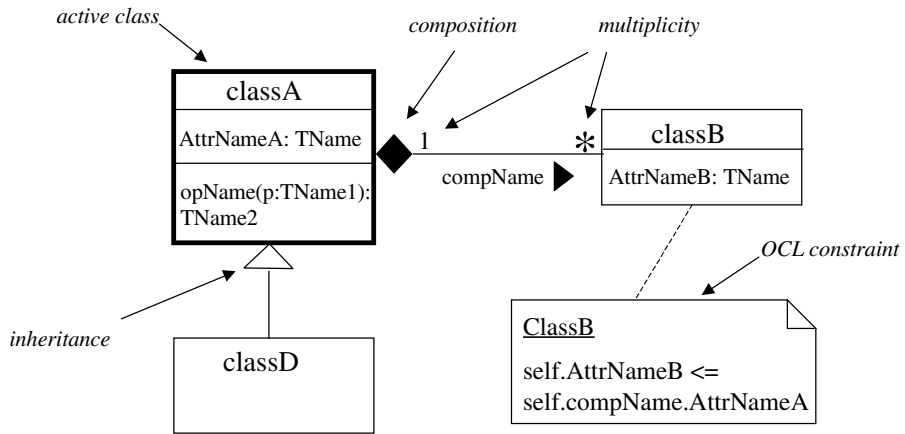
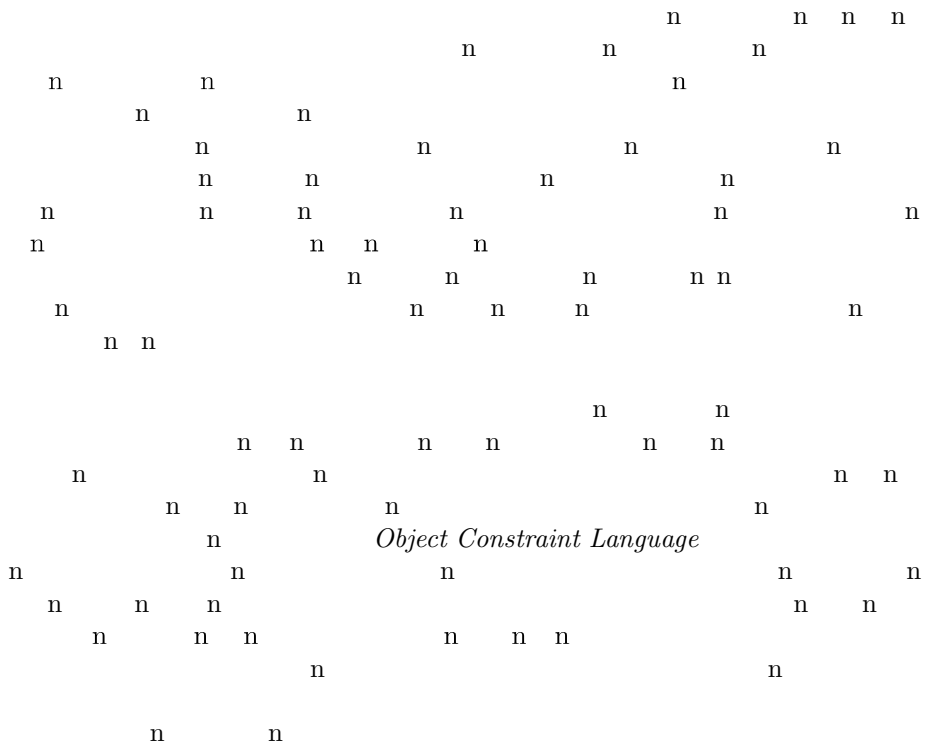


Figure 2.

Modeling the Structural Aspect.

class n object diagrams,



Modeling the Behavioral Aspect.

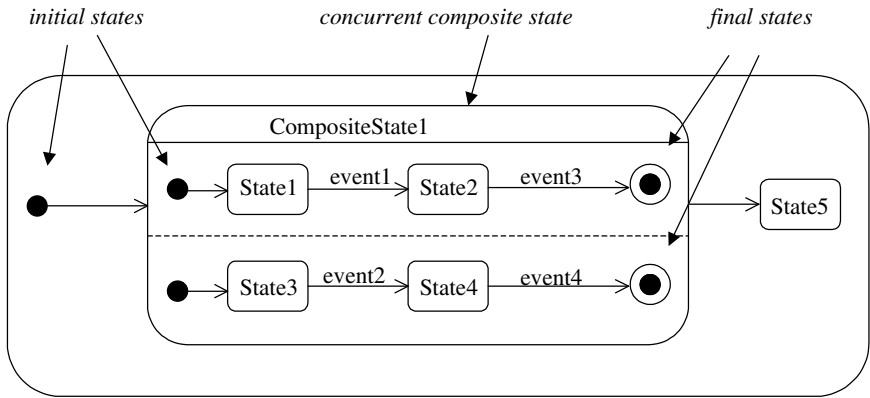
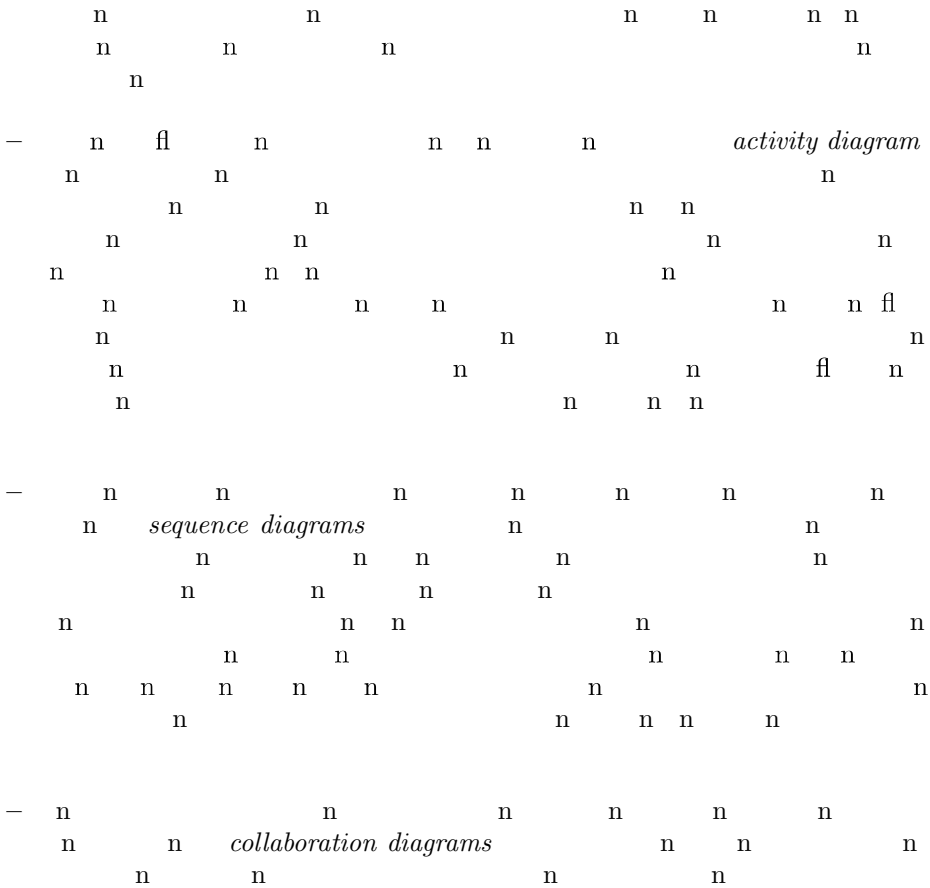


Figure 4. n



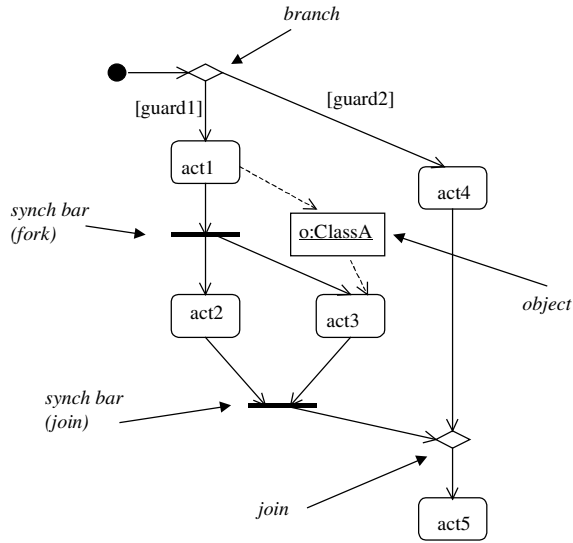
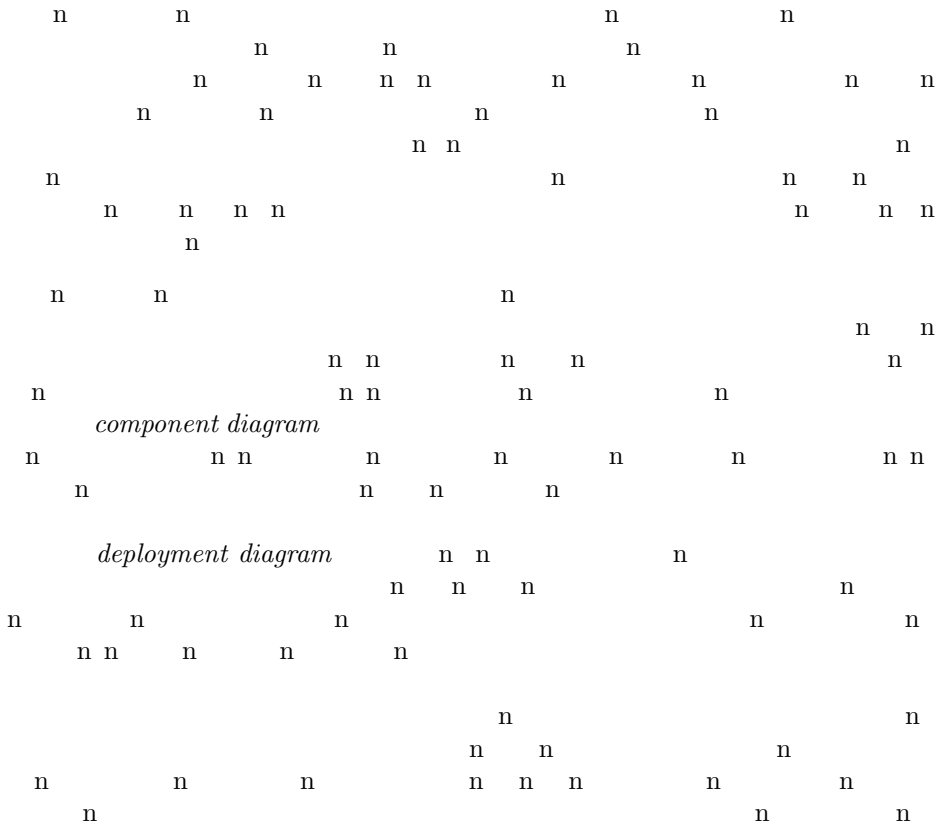


Figure 5.



```

sequenceDiagram
    participant o4 as o4:ClassB
    participant o1 as o1:ClassA
    participant o2 as o2:ClassB
    participant o3 as o3:ClassD

    o4->>o1: opB2
    activate o1
    o1->>o2: opB1
    activate o2
    o2->>o3: opD
    activate o3
    o3-->>o1: return message
    deactivate o3
    deactivate o2
    deactivate o1
  
```

3 Language Definition

[illegible]

[illegible][illegible]

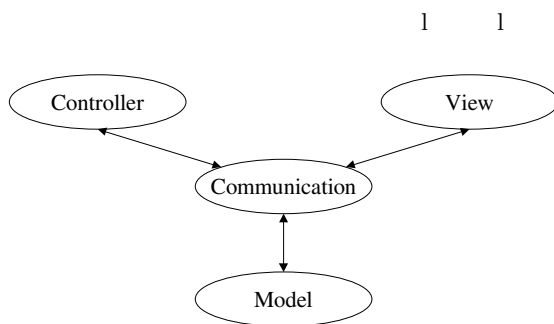


Figure 8.

The diagram is a collection of small, scattered characters and symbols. The characters include the letter 'n', the letter 'm', the letter 'f', and a horizontal line symbol '—'. These are distributed across the page in a non-uniform, abstract pattern. Some characters are grouped together, while others are isolated. The overall effect is that of a random or semi-random arrangement of these specific symbols.

	l	k	k l				
	l			l	l	l	
k						l	
		k					l
	l						
	l						
					ll	l	l
					l		l

LINKS

n n

Verification of Timed and Hybrid Systems

1

BRICS*, Dep. of Computer Science, Aalborg University, Denmark
kgl@cs.auc.dk

Abstract. UPPAAL [UPP, BLL⁺98] is an integrated tool environment for modelling, simulating and verification of real-time and hybrid systems, developed jointly by BRICS at Aalborg University in Denmark and by DoCS at Uppsala University in Sweden. In this talk we will review the status of the currently distributed version of UPPAAL and describe in more detail the ongoing developments which are to be incorporated in future releases of the tool.

Extended Modelling Language

[illegible]

* BRICS: Center for Basic Research in Computer Science at Aarhus and Aalborg University

Beyond Model-Checking

	1	UPPAAL	scheduling	
1	(ll	k	1 ll
	1 1	1		ll
		ll	UPPAAL	1 1
1	1	optimal	1	1
1	1 1		1	(
ll	1 1	1 1	UPPAAL	
	1 1	1		ll

Improvement of Verifier

	UPPAAL	1	1	
	1 1	ll	Clock Difference Di-	
agrams	+	1 k	(ll	1
	1	1		UPPAAL
	1	ll	1	
		1	1	
		1		
	1	1	1	
	1 1	1	ll	ll
1	ll 1			1
1	1			
		ll 1	UPPAAL	1
	1	1	1	
	1	1	1	
		1	abstraction	composi-
tionality	ll		UPPAAL	

¹ in the sence that model-checking a single timed automaton is either EXPTIME- or PSPACE-complet depending on the expressiveness of the logic considered

- [BJLY98] Johan Bentsson, Bengt Jonsson, Johan Lilius, and Wang Yi. Partial Order Reductions for Timed systems. *Lecture Notes in Computer Science*, 1998.
- [BLL⁺98] Johan Bengtsson, Kim G. Larsen, Fredrik Larsson, Paul Pettersson, Yi Wang, and Carsten Weise. New Generation of UPPAAL. In *Int. Workshop on Software Tools for Technology Transfer*, June 1998.
- [BLP⁺99] G. Behrmann, K.G. Larsen, J. Pearson, C. Weise, and W. Yi. Efficient Timed Reachability Analysis Using Clock Difference Diagrams. *Lecture Notes in Computer Science*, 1633, 1999. In Proceedings of Computer Aided Verification 1999.
- [BS99] René Boel and Geert Stremersch. VHS case study 5: modelling and verification of scheduling for steel plant at SIDMAR. draft, 1999.
- [CL00] Franck Cassez and Kim G. Larsen. The Impressive Power of Stopwatch Automata. Submitted for publication., 2000.
- [Feh99] Ansgar Fehnker. Scheduling a steel plant with timed automata. Technical Report CSI-R9910, Computing Science Institute Nijmegen, 1999.
- [Hen96] T. Henzinger. The theory of hybrid automata. In *Proceedings of 11th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1996.
- [HLP00] Thomas Hune, Kim G. Larsen, and Paul Pettersson. Guided Synthesis of Control Programs Using UPPAAL. *To appear in Proceedings of Workshop on Distributed Systems Verification and Validation*, 2000.
- [HLS99] Klaus Havelund, Kim G. Larsen, and Arne Skou. Formal Verification of a Power Controller Using the Real-Time Model Checker UPPAAL. *Lecture Notes in Computer Science*, 1601, 1999. In proceedings of 5th International AMAST Workshop, ARTS'99.
- [Hun99] Thomas Hune. Modelling a Real-Time Language. In *proceedings of 4th Workshop on Formal Methods for Industrial Critical Systems, FMICS'99.*, 1999.
- [IKL⁺] Torsten K. Iversen, Kre J. Kristoffersen, Kim G. Larsen, Morten Laursen, Rune G. Madsen, Steffen K. Mortensen, Paul Pettersson, and Chris B. Thomasen. Model-Checking Real-Time Control Programs. To be published in Proceedings of Euromicro 2000.
- [JLS00] Henrik E. Jensen, Kim G. Larsen, and Arne Skou. Scaling Up UPPAAL - Automatic Verification of Real-Timed Systems Using Compositionality and Abstraction. Submitted for publication, 2000.
- [KLPW99] K. Kristoffersen, K. Larsen, P. Pettersson, and C. Weise. Experimental batch plant - VHS case study 1 using timed automata and UPPAAL. BRICS, University of Aalborg, Denmark, May 1999.
- [LPL99] Kim G. Larsen, Paul Pettersson, and Hans Henrik Løvengreen. Real-time systems — course, 1999. <http://www.cs.auc.dk/~kgl/DTU00/Plan.html>.
- [LWYP99] Kim G. Larsen, Carsten Weise, Wang Yi, and Justing Pearson. Clock Difference Diagrams (extended version). *Nordic Journal of Computing*, 6, 1999.
- [MLAH99] J. Møller, J. Lichtenberg, H. R. Andersen, and H. Hulgaard. Fully symbolic model checking of timed systems using difference decision diagrams. In *Workshop on Symbolic Model Checking*, The IT University of Copenhagen, Denmark, June 1999.
- [UPP] The UPPAAL home page. <http://www.uppaal.com>.
- [VHS] The VHS project home page. <http://www-verimag.imag.fr/VHS>.

Parametric Stochastic Well-Formed Nets and Compositional Modelling*

l ll n¹ u nn n ll¹ n ul n n n²

¹ Dip.to di Informatica, Università di Torino, Italy
susi@di.unito.it

² DSTA, Università del Piemonte Orientale, Alessandria, Italy
giuliana@di.unito.it

Abstract. Colored nets have been recognized as a powerful modelling paradigm for the validation and evaluation of systems, both in terms of compact representation and aggregate state space generation. In this paper we discuss the issue of adding compositionality to a class of stochastic colored nets named Stochastic Well-formed Nets, in order to increase modularity and reuse of the modelling efforts. This requires the notion of Parametric Stochastic Well-formed net: nets in which a certain amount of information is left unspecified, and is instantiated only upon model composition. The choice of the compositional rule has been based on previous work on layered models for integrated hardware and software systems (the processes, services and resources methodology), and an example of layered modelling with Parametric Stochastic Well-formed net is presented to show the efficacy of the proposed formalism.

1 Introduction and Motivations

n n n n u l l l u n
l n n n n u n lu n
n n u l n n n u nn n
l l l u ul ll n ll l u
u n n n u n l
ul l n n n l u n
n l n l n u l l l
n u n l n 2 n lu n n
n u ll l u ul u n
n l n n u l
l n ll n n
n l n n n u l 2
n l n n
n u l u ll n l
n n l l n n n n
n l ll 1) n n n n n

* We acknowledge contribution of the EEC project 28620 TIRAN.

l 2) u l l u kn n l l
n n n l 3) l
l n n u n l n l 4)
l l n n n n
n l n l n n
l n n l n n
l l n un n n l l n
un n l n n n l ul ll n
l l n l lu
l n u n n l u n
n l n l l l l
l u u n un n
n n u fl l l l
n u n n l n n
n n n u l n l ul n n n l
l n n n l ul n n n
l n n lu n n n
n k n n n l u

2 PSR Methodology

l n n n u u n ul n
n l n levels ll n u
ll n l n u
ll l l ll n l l l ul u n
l l ll n ll l l u n n nl
n n l l

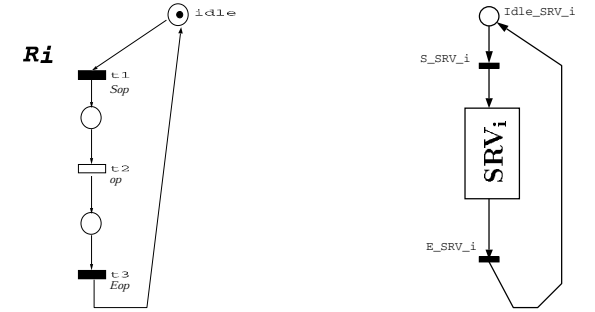


Fig. 1. u n op n srv

[illegible]

3 PSWN Definition

n n n n u n l ll n
 l l n n n ul u l u kl k
 l l l n n l l ul

u l n u n u l n u
u k l n l u l l u n
n n l n n l
l l n l n u n u l
n u l u n n n l n
u n l n n u n l l n
n l n n n u
n l l l n l l
n n l fl ll
l u n n n n n
ll n ll n

3.1 A Quick Reminder of SWN

n n n u u n n l n
basic color classes C_1, \dots, C_n l l C_i n n
n l l l l
l u l l n l
n n l *static subclasses* C_i $C_{i,1} \dots C_{i,n_i}$ j, k j
 $k, C_{i,j} \cap C_{i,k}$ l l n n n u l n
u l n u l
l l ul n n n u l
n n n n *fast* n n n n *slow* n 2
n A, B, C n n u n ll
k A a_1, a_2 B b_1, b_2 n C c_1, c_2
l *color domains* n n u n
u l l l l l n n 2 2 l p_1
l n A l l l l PP l n A B C
l l u n n n u u u u
n n *variables* l n u l u u l
l l l n u l n
l n l nn n n 2 2
u n n p_1 t_1 l x l A l u n n
 h, i, j n u l PP l l l A B n
C l n u n - u n n n
n u u n l n u n u n
 S_{sc} n k n l u l l sc
n n n n n l l u l
l A B C l n k n l n n u l
 t_1
l n u n u u n n
n u n l l x l n n u p_4
n u n l l

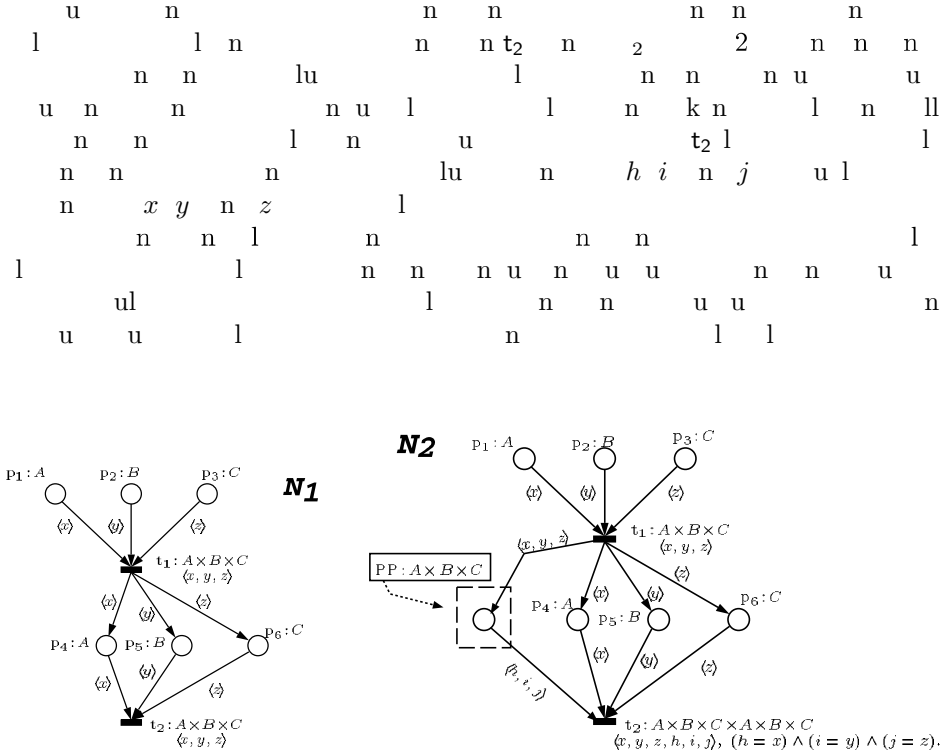


Fig. 2.

3.2 A Motivation for PSWN: The WRONG-MATCH Problem

u u n 1 2 n l kn M_0 n
 kn l l $p_1, p_2,$ n p_3 n n n ll n
 n u n M_0 $t_1, \underline{a_1, b_1, c_1}$ M_1 $t_1, \underline{a_2, b_2, c_2}$ M_2 ll kn
 n u l t_1 n n kn l l $p_4, p_5,$ n p_6
 n n n n t_2 n l n M_2 n ll t_2, x, y, z u
 x, y, z M_2 p_4 M_2 p_5 M_2 p_6

E M_2, t_2 t_2, a_1, b_1, c_1 , t_2, a_1, b_1, c_2 , t_2, a_1, b_2, c_1 ,
 t_2, a_1, b_2, c_2 , t_2, a_2, b_1, c_1 , t_2, a_2, b_1, c_2 ,
 t_2, a_2, b_2, c_1 , t_2, a_2, b_2, c_2 .

n n A n B u C t_1
 n u n n t_2 n n n n
 ll u n n a u u b n c
 u n k u b n c_2 n n a_2

$$\begin{array}{cccccccccccccccc}
 u & & u & b_2 & n & & c_2 & & ll & & & & l & & & & \\
 & l & lu & n & & n & n & n & 2 & & 2 & n & l & PP & u & & k \\
 k & & l & & & u & & & ll & & & & n & l & & & \\
 n & & u & & & & & l & n & & u & u & n & n & n & & \\
 u & n & n & n & t_2 & n & u & & & & n & u & & & n & t_2 \\
 & & & & & t_1 & n & & & & & & & & & &
 \end{array}$$

$$E \ M_2, t_2 \quad t_2, a_1, b_1, c_1, a_1, b_1, c_1 \quad , \quad t_2, a_2, b_2, c_2, a_2, b_2, c_2 \quad .$$

$$\begin{array}{cccccccccccccccccccc}
 lu & n & & & k & n & u & it \ lacks \ compositionality & & l & & & & & & & \\
 & u & & & ul & & & n & & l & n & & & & & & \\
 l & l & n & n & & u & l & l & n & & ul & PP & l & n & & & \\
 & & & l & PP & & & & & & n & n & & l & & & \\
 & & l & PP & l & n & & PP & l & n & & & & u & & & \\
 l & l & kn & l & & & & u & n & & n & ul & n & & n & l & \\
 l & & & u & n & n & n & l & l & n & & l & & n & n & & \\
 n & & n & n & & & & ul & & n & PP & l & & & & & \\
 n & A & B & C & & u & l & l & kn & & u & & l & l & l & & \\
 & & l & l & kn & & u & & u & l & l & l & n & & & & \\
 l & l & & should \ not \ be \ aware \ of \ the \ color \ class \ and \ on \ the \ color \ class \ structure & & & & & & & & & & & & & \\
 & u & & l & l & n & & u & & u & & & & & & & \\
 & u & & ul & n & & l & & n & u & n & n & & & & & \\
 ll & l & l & un & n & & ll & n & n & & n & l & l & & & & \\
 & & n & n & l & ll & u & & n & unkn & n & u & & & & & \\
 l & l & n & unkn & n & & l & l & n & & & n & & & & & \\
 ul & k & & n & n & n & un & n & l & & & l & l & & & & \\
 un & n & l & l & & ll & & ll & & & & & & & & &
 \end{array}$$

3.3 Parametric SWN Definition

$$\begin{array}{cccccccccccccccc}
 cd \ t & & l & & n & & n & n \ t & & C_i, x & & & & & & & \\
 x & & l & t & n & C_i & & l & l & n & n & n & & & & &
 \end{array}$$

Definition 1 (Parametric Stochastic Well-formed Nets). A

ll (PSWN) is a nine-tuple:

$P, T, \text{Pre}, \text{Post}, \text{Inh}, \text{pri}, \text{w}, cd, w$

- $P, T, \text{Pre}, \text{Post}, \text{Inh}, \text{pri}$, and w , (that stay for places, transitions, input, output, inhibitor, and priority functions, and weight assignment respectively) are defined as for classical SWN,
- C_1, \dots, C_n , where C_1, \dots, C_n is the finite set of finite color classes, $(i, C_i \text{ } c_{i_1}, \dots, c_{i_{n_i}})$ and XC_1, \dots, XC_n is the finite set of color classes
- cd defines the color domain of places and transitions, with $cd \ P^n, n$, and $cd \ T \ (VAR^m, m)$, where VAR is the set of variables.

$$\begin{array}{cccccccccccc}
 & & & & n & & l & l & & n & & ull & n \\
 & & un & n & & n & & k & n & n & & n & l & n & n & n \\
 un & n & l & & ll & n & n & & u & n & & n & & n & & n \\
 u & & n & n & n & n & u & ll & l & & n & & n & l & l \\
 & n & & un & n & n & & l & l & & n & n & n & & \\
 n & & un & n & & & l & l & & n & n & & & \\
 & l & l & n & l & lu & & & & & & & & &
 \end{array}$$

Definition 2 (Multilabeled PSWN). A $(MPSWN)$ is a twelve-tuple:

$$P, T, \text{Pre}, \text{Post}, \text{Inh}, \text{pri}, \tau, cd, w, \lambda$$

where:

- $P, T, \text{Pre}, \text{Post}, \text{Inh}, \text{pri}, \tau, cd$ and w , are defined as for PSWN,
- λ is a labeling function. If L is a set of labels, then the labeling is defined as $\lambda : T \rightarrow \mathcal{P}(L)$, where $\mathcal{P}(L)$ is the powerset of L ,
- $\text{imp}(t, l)$, with $l \in \lambda(t)$ is a pair (C, x) of $cd(t)$, or the empty pair $(-, -)$, and it is named “import function” of t over label l , and
- $\text{exp}(t, l)$, with $l \in \lambda(t)$ is a list of pairs (C_i, x_i) , with $C_i, x_i \in cd(t)$, or the empty pair $(-, -)$, and is named “export function” of t over label l ,

$$\begin{array}{cccccccccccc}
 & & n & n & l & l & n & & n & & n & l & n & n \\
 n & l & l & & & & & & n & n & & & & l \\
 n & & n & & & & & & & & & & &
 \end{array}$$

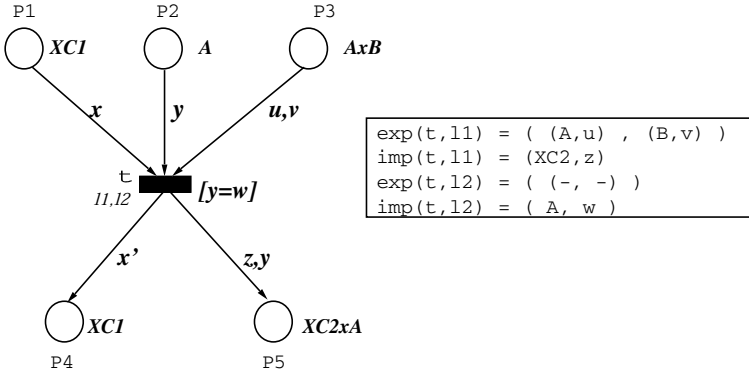
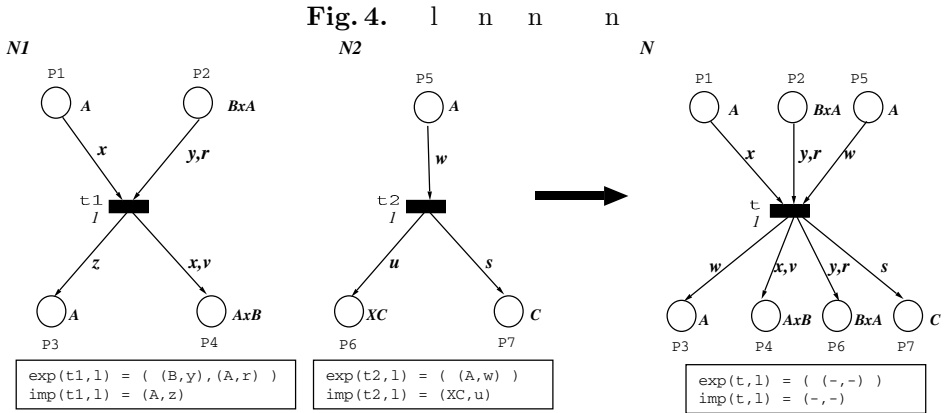
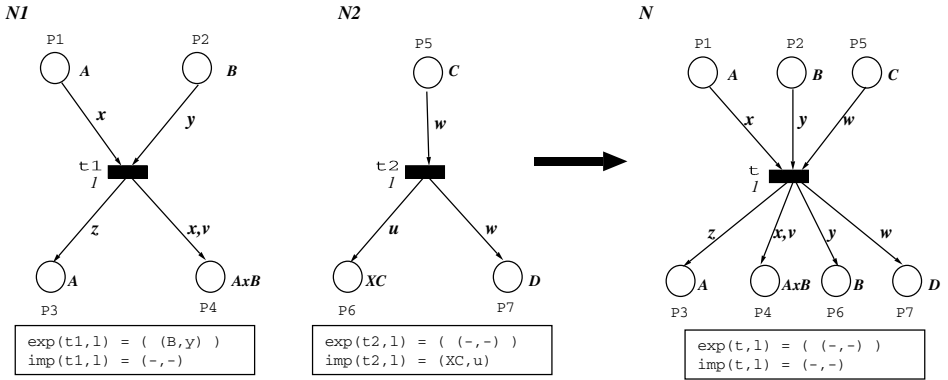


Fig. 3. $(MPSWN)$

$$\begin{array}{cccccccccccc}
 & & & & n & & lu & n & l & & n & n \\
 l & n & l & ul & u & & n & un & & l & & l \\
 & & & l & l & ul & u & & l & l & n & n \\
 n & n & u & & l & n & & n & un & n & ul &
 \end{array}$$

— n n l ll u n n lu
 — n n n l u n l n
 n
 u n u n l ul n l
 1 n 2 u n l l l n
 N l B n l y n N2 l
 XC n l u u n n u u l
 P6 n l B n l n t P6 n y
 n n n l n n n l n N n N2
 n n n l n u n n k u n
 n l u n n n n
 n n n n ll u the sets of the
 names of the variables of the nets that are composed are disjoint



l n n l l u l
 B A n lu y, r u l u u l z XC l
 w ul n n n nt n l $P3$ kn
 l kn $P5$
 u n nnn l ln n 2
 n n l l l nu $t1$ n 1
 n n $t1-2$ n $t1-3$ l B n n $t1$ l
 $XC2$ $P10$ B n lu y n n
 n n $t1$ l XC n n n l ll
 n $N2$ n l n l $P3$ l l
 $P3-C$ n n l C XC u $t2$ n l
 $P3-E$ n n l E XC u $t3$
 n n l l $P3$ n 1 n nn un n
 l XC n l n n n n l l XC
 nnu n u u n l llu n ul
 l n l n n nk n n **C2-**
propagation n n n n l n ll
 n n l n n l n n
 kn n n n n l l
 n u l l XC n ll l
 n n n n un l n l un l
 n n ll n n l n
 n l l n n n
 n l l n n n l n
 ll u n n n 1 n u u $l1, l2$
 ul n ul 1 k n l $P3$ n u u
 u $l1$ n n l $P4$ u u u $l2$
 u 2 l C n E n n u
 l w n r n l ul n n nn
 u n N n $N2$ ul n n
 n u $t1$ $t2$ n l $l1$ n u ul n
 n $t3$ $l2$ nk n n **C2-import**

4.1 Composition of Two MPSWN

n ll n n l l n k n n u
 l n un n n n n 1
 n n n 2 n n n n
 ul l ln l l nnn n l u
 n l un

Definition 3 (Composition $_E$). Let's take two LPSWN with injective labeling i $P_i, T_i, \text{Pre}_i, \text{Post}_i, \text{Inh}_i, \text{pri}_i, i, cd_i, w_i, i, i, \lambda_i$ with i

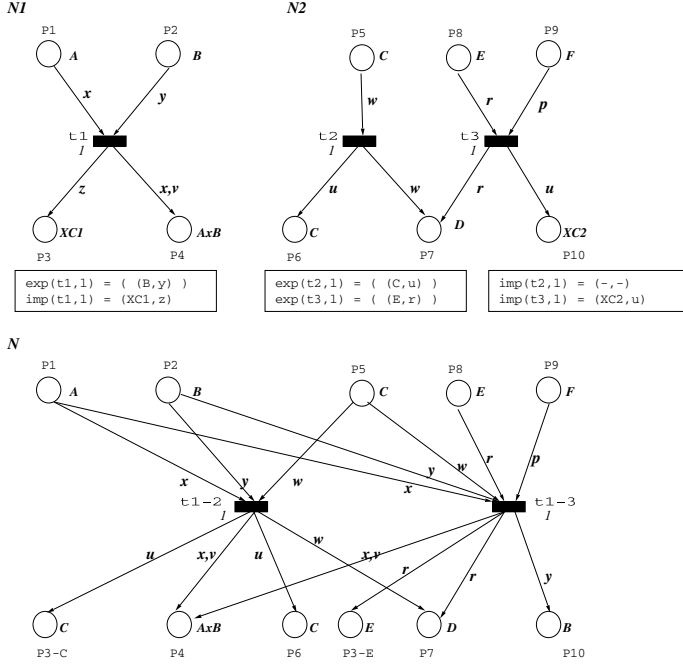


Fig. 6.

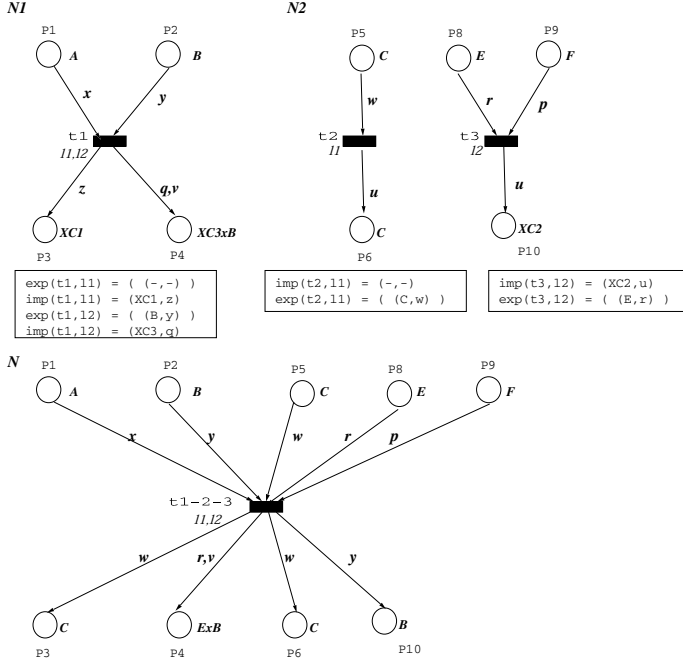
 $n \quad n \quad n \quad l \quad l \quad n$


Fig. 7.

 $ul \quad l \quad l \quad n$

, 2 . The LPSWN $\lambda_1 T_1 \lambda_2 T_2$ resulting from the composition of λ_1 and λ_2 on the set of labels E is the following LPSWN:

$$P, T, \text{Pre}, \text{Post}, \text{Inh}, \text{pri}, \text{cd}, \text{w}, \text{ , } \text{ , } \lambda$$

with: $P = P_1 \cup P_2$, $T = T_1 \cup T_1^E \cup T_2$ (all transitions of T that do not synchronize, plus all that of T_2 where T_i^E is the subset of transitions $t \in T_i$ such that $\lambda(t) \in E$);

$\text{cd}_1 = \text{cd}_1^E \cup \text{cd}_2^E$ where cd_i^E is the subset of cd_i that is instantiated to basic color classes thanks to import operations on the labels of E ;

$\text{cd}_2 = \text{cd}_2^E$, where cd_i^E represents the restriction of cd_i to the subset of labels E and $\text{cd}_1 = \text{cd}_1^E \cup \text{cd}_2^E$, where cd_i^E represents the restriction of cd_i to the subset of labels E , moreover

$$\text{cd}(t) = \begin{cases} \text{cd}_1(t) & \text{if } t \in T_1 \cup T_1^E \\ \text{cd}_2(t) & \text{if } t \in T_2 \cup T_2^E \end{cases}$$

$$\text{cd}_1(t) = \begin{cases} \text{cd}_1(t) & \text{if } t \in T_1 \cup T_1^E \\ \text{cd}_2(t) & \text{if } t \in T_2 \cup T_2^E \end{cases}$$

where $\text{cd}_i(t, l)$ represents the substitutions due to the unification of color classes and variables caused by the import of i and the export of j , with $i \neq j$;

$$\text{cd}(p) = \begin{cases} \text{cd}_1(p) & \text{if } p \in P_1 \\ \text{cd}_2(p) & \text{if } p \in P_2 \end{cases}$$

where $\text{cd}_i(p)$ is $\text{cd}_i(p)$ where all parametric classes in cd_i^E have been substituted by the corresponding basic color classes of $\text{cd}_j, i \neq j$ imported through L^1 ; for $F_i = \text{Pre}_i, \text{Post}_i, \text{Inh}_i$, we define:

$$F(t) = \begin{cases} F_1(t) & \text{if } t \in T_1 \cup T_1^E \\ F_2(t) & \text{if } t \in T_2 \cup T_2^E \\ F_1(t) \cup F_2(t) & \text{if } t \in T_2^E \end{cases}$$

and, again, F_i stands for F_i where variables of colors belonging to cd_i^E have been substituted by the imported variables.

$$\begin{matrix} n & n & & n & n & & n & & ul & l & l & n & u \\ & & n & & u & nl & k & n & un & & n & n & n & n \\ & & & n & ul & & & n & & n & & n & n \\ l & & n & n & n & n & n & l & n & l & u & n & u & ll & l \\ & l & & n & n & l & n & n & l & n & n & n & n & n \\ u & l & & un & & n & l & & u & & & l & l \end{matrix}$$

¹ Observe that this operation is well defined since, due to injectivity of the labeling function and to the constraints imposed on PSWN, each parametric color is instantiated to exactly one basic color class

5 Example of PSWN Use

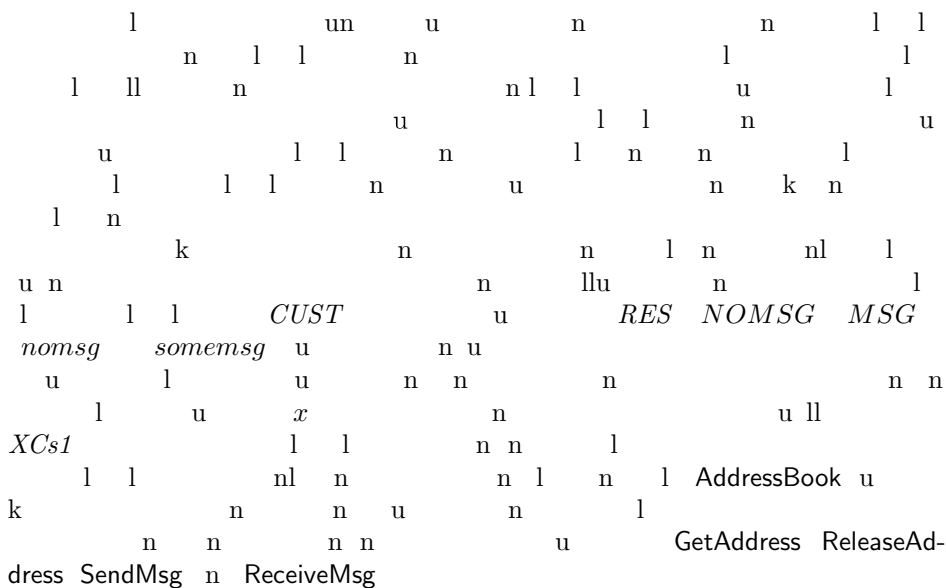
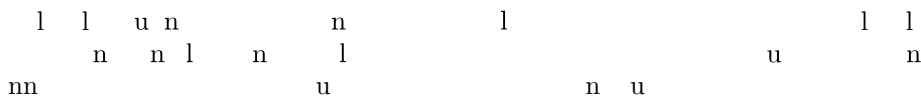


Fig. 8.

5.1 Level \mathcal{S} : The Services



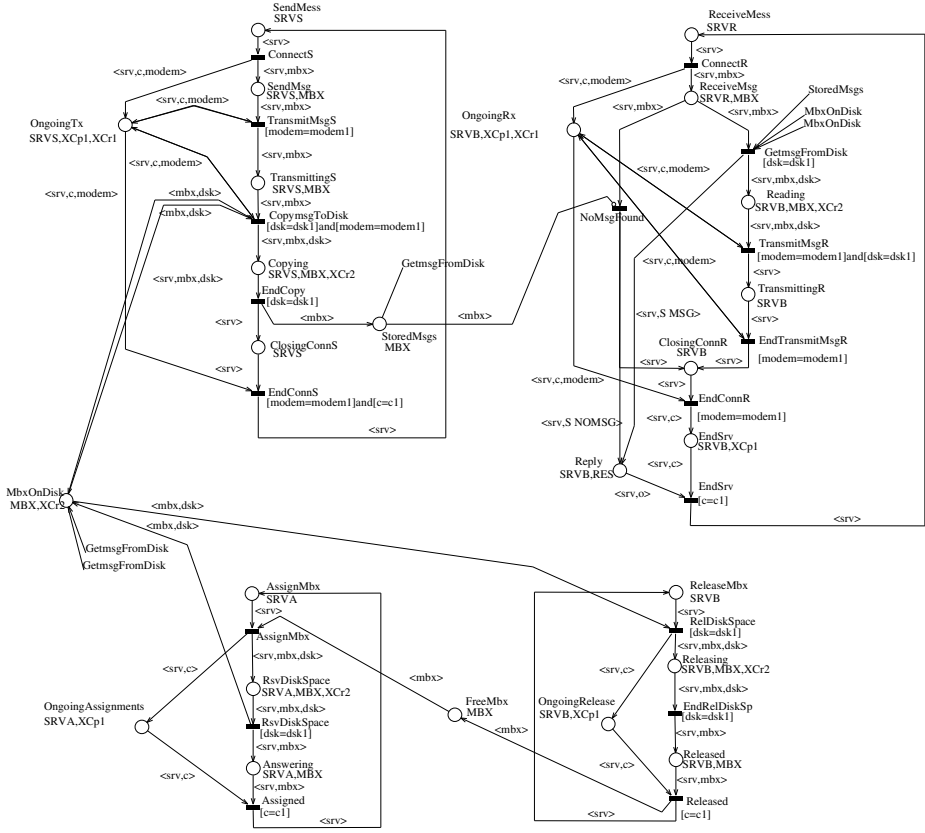


Fig. 9.

1 1

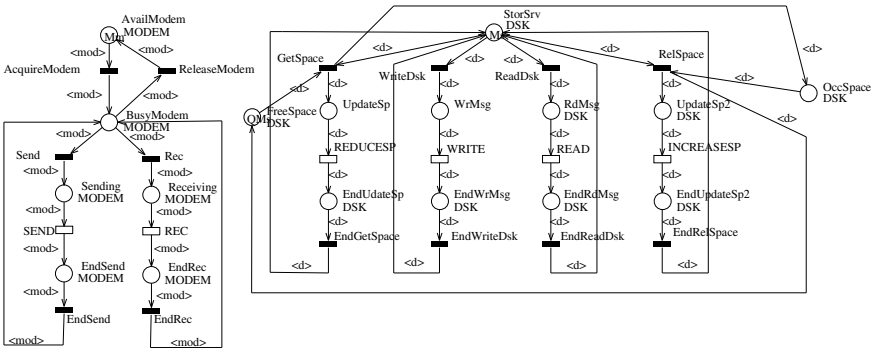


Fig. 10.

u 1 1

l n n u u nn
 2 l l u u
 un n n n n n
 l l l l n n *logical resources*
 n u l n n l
 n n l *Assign Mailbox* n l l u
 n *Release Mailbox* n l u n *Send Message*
 n l u u n n ll *Receive Messages* n
 u u n n n l n k
 n l n n
 k n n l l k k l k
 n n l l n n
 k l *Send Message*
 u nn n l
 n n u n
 n k n n l n ll
 nn n l l *StoredMsg* u k k n
 l n n l l *Receive Messages*
 nn n ll
 k n l n n
 n l n n n
 l n u n n n
 l k n un l l *RES* l
 n n lu n *the value of variable o* l l
 n n *EndSrv* l l ll n n n
 l l
 l u l n n l l *MBX* l n
SRVS n n
 nk l n *SRVR*
 nk l n *SRVA*
 n n n l n *SRVE*
 l l n u n *RES* n l n
 l l n u nu l
XCp1: u k k n u n
 kn l
 n l n u n lu n n l l n
 u n n lu n l n l u n
 OngoingTx OngoingRx EndSrv OngoingAssignment n OngoingRelease
XCp1: u k k n u n l
 u n n n n l
 l n l OngoingTx OngoingRx
XCp2: u k k n k
 n l l n

l n lu n l n l u n MbxOnDisk u k
k n l k u n l l l
n n n un l l RsvDiskSpace Releasing Copying
n Reading ll n n n n k n
u

offered labels n n n l l n l l
n n n un n u n
n 2 nn n l l l n n lu n
n n n n n n u n ul n l
n n u n n n l l

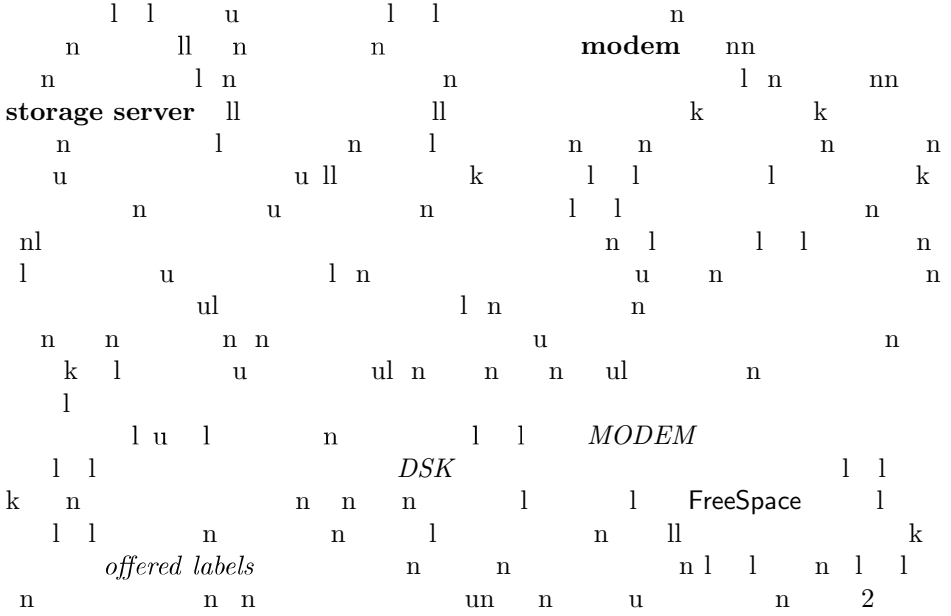
	• •	• /	• /
- -		• • • • • • • • • • • • • • •	• • • • • • • • • • • • • • •
- -		• • • • • • • • • • • • • • •	• • • • • • • • • • • • • • •
- -		• • • • • • • • • • • • • • •	• • • • • • • • • • • • • • •
- -		• • • • • • • • • • • • • • •	• • • • • • • • • • • • • • •

Table 1. n n l l n

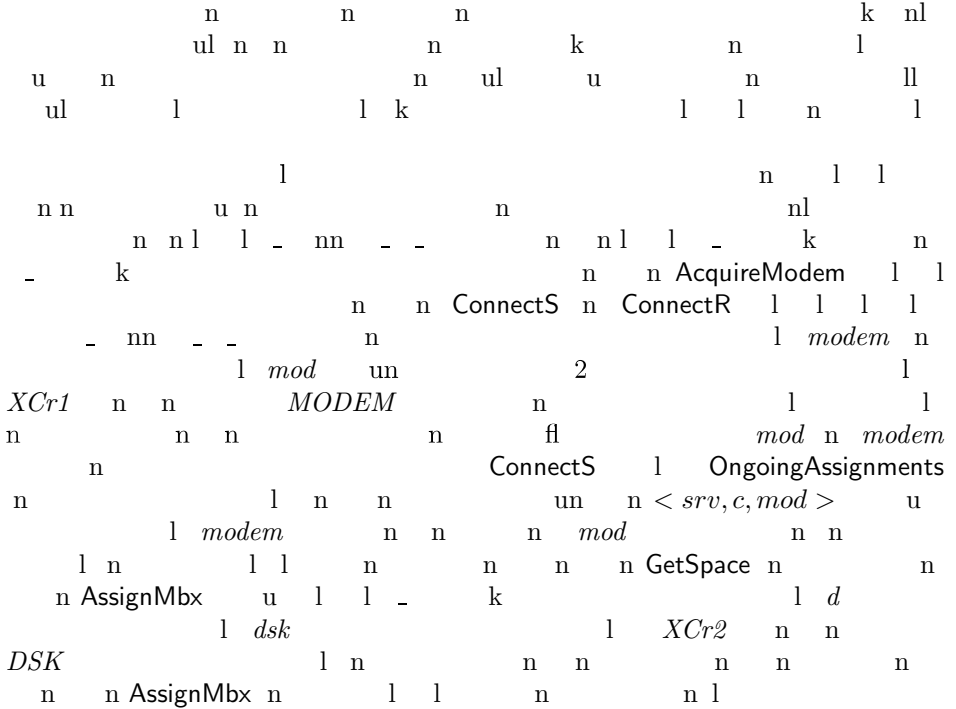
	• •	• /	• /
- - - - - -		• • • • • • • • • • • • • • •	• • • • • • • • • • • • • • •
- -		• • • • • • • • • •	• • • • • • • • • •
- -		• • • • • • • • • •	• • • • • • • • • •
- -		• • • • • • • • • •	• • • • • • • • • •
- -		• • • • • • • • • •	• • • • • • • • • •
- -		• • • • • • • • • •	• • • • • • • • • •
- -		• • • • • • • • • •	• • • • • • • • • •

Table 2. n n l l n

5.2 Level \mathcal{R} : The Resources Level



5.3 Composition



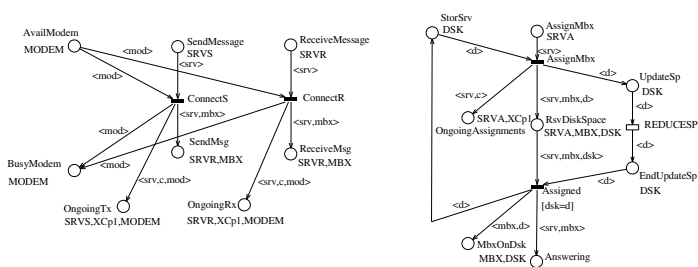
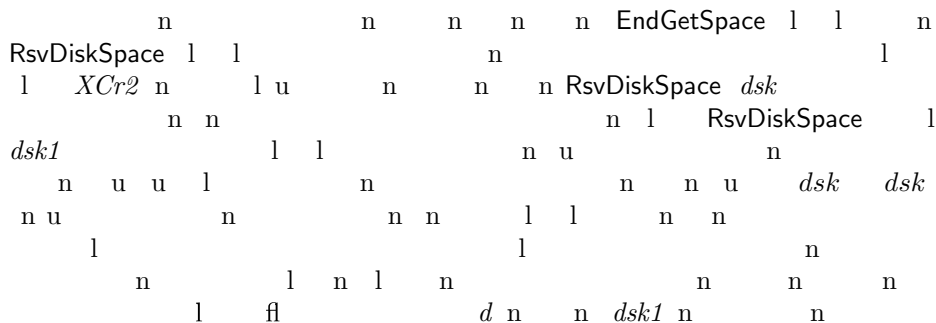


Fig. 11.

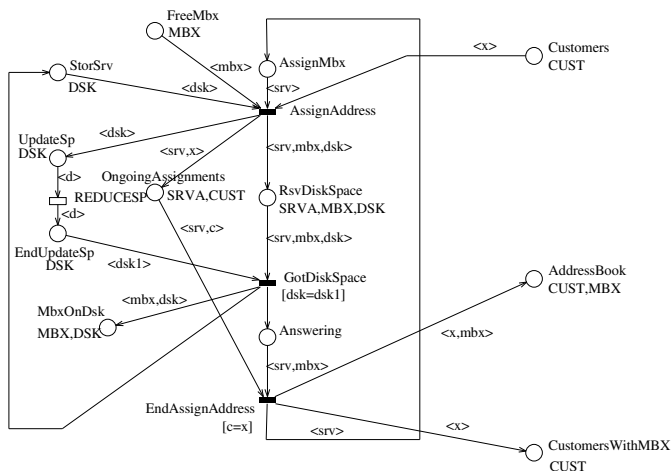


Fig. 12.

n ll fl l l ll l k l k l
 u n n GetAddress n n l l
 u n 2 ll GetAddress un l
 n n n n l l n n
 GetAddress1 l l - n GetAddress2
 l l - n n u n
 x n n n CUST l l l n
 n n u n n u
 l n l n
 n u 2 ul n

6 Conclusions

n n n n l ll
 l l n l un n
 n l n n n u n n ll
 l l l n n l l u u
 l l u n n l
 n l n u u
 l n n l ul l ll n n n
 l n n l ul n n l
 ul n n n u n l l n k
 u n k n l n n n
 ul n n n n n
 n n n l n 2 n
 n n n n l 2 2 u
 n 2
 u u k ll ll l n n n n
 l u n n l n n
 n n n n ll
 un n n n nl n l n
 n n n l n n n u n l
 l l n ll l l n l n
 n l n l
 ul n n l
 n l n un n n *single* u u l
 l
 n n n n u n ll n
 n k n n l n n un n w n
 un n **pri** l l n l n k n
 l n n n n l ul n
 k un l n u n
 n n n n u n

References

- [1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. • • • • • J. Wiley, 1995.
- [2] C. Anglano, S. Donatelli, and R. Gaeta. Parallel architectures with regular structure: a case study in modelling using SWN. In • • • • • 5th • • • • • , Toulouse, France, October 1993. IEEE-CS Press.
- [3] E. Battiston, O. Botti, E. Crivelli, and F. De Cindio. An incremental specification of a hydroelectric power plant control system using a class of modular algebraic nets. In • • • • • 16th • • • • • , Torino, Italy, 1995. Springer Verlag. Volume LNCS935.
- [4] E. Best, H. Flrishhacl ANF W. Fraczak, R. Hopkins, H. Klaudel, and E. Pelz. A class of composable high level Petri nets with an application to the semantics of $B(PN)^2$. In • • • • • 16th • • • • • , Torino, Italy, 1995. Springer Verlag. Volume LNCS935.
- [5] P. Buchholz. A hierarchical view of GCSN and its impact on qualitative and quantitative analysis. • • • • • , (15), July 1992.
- [6] G. Chiola, C. Dutheillett, G. Franceschinis, and S. Haddad. On Well-Formed coloured nets and their symbolic reachability graph. In • • • • • 11th • • • • • , Paris, France, June 1990. Reprinted in • • • • • , K. Jensen and G. Rozenberg (editors), Springer Verlag, 1991.
- [7] G. Chiola, C. Dutheillett, G. Franceschinis, and S. Haddad. A Symbolic Reachability Graph for Coloured Petri Nets. • • • • • , 176(1&2):39–65, April 1997.
- [8] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. GreatSPN 1.7: Graphical Editor and Analyzer for Timed and Stochastic Petri Nets. • • • • • , 24(1&2):47–68, November 1995.
- [9] S. Christensen and L. Petrucci. Modular state space analysis of coloured Petri nets. In • • • • • 16th • • • • • , Torino, Italy, 1995. Springer Verlag. Volume LNCS935.
- [10] S. Donatelli and G. Franceschinis. The PSR methodology: integrating hardware and software models. In • • • • • 17th • • • • • , Osaka, Japan, june 1996. Springer Verlag. LNCS, Vol 1091.
- [11] S. Haddad and P. Moreaux. Evaluation of high level Petri nets by means of aggregation and decomposition. In • • • • • 6th • • • • • , Durham, North Carolina, U.S.A, October 1995.
- [12] K. Jensen. • • • • • Springer Verlag, 1992.
- [13] K. Jensen. • • • • • Springer Verlag, 1995.
- [14] Isabel C. Rojas M. • • • • • . PhD thesis, University of Edinburgh, 1997.
- [15] E. Teruel, G. Franceschinis, and M. De Pierro. Clarifying the priority specification of gspn: Detached priorities. In • • • • • 8th • • • • • , Zaragoza, Spain, September 1999. IEEE-CS Press.

Reducing k -Safe Petri Nets to Pomset-Equivalent 1-Safe Petri Nets

$\mathcal{L}(N) = \{ \sigma \in \Sigma^* \mid \exists m \in \mathbb{N}^n. m \cdot N \cdot \sigma = 0 \}$

Abstract.

We show that for any k -safe Petri net N , there exists a 1-safe Petri net N' such that $\mathcal{L}(N) = \mathcal{L}(N')$. The construction of N' is based on the decomposition of the places of N into two parts, one of which is used to store the state of the net, and the other is used to store the state of the net modulo k .

Keywords: Petri nets, Pomsets, Safety, Decomposition

1 Introduction

Petri nets are a powerful tool for modeling concurrent systems. They consist of a set of places and a set of transitions. Places are represented by circles and transitions by rectangles. Places contain tokens, which are represented by small circles. Transitions are labeled with actions, which are represented by letters. The state of a Petri net is determined by the number of tokens in each place. The set of all possible states of a Petri net is called its state space. The set of all possible sequences of actions that can be performed on a Petri net is called its language.

2 Nets and Their Pomset Languages

The pomset language of a Petri net is the set of all possible sequences of actions that can be performed on the net, where the order of the actions is preserved. The pomset language of a Petri net is denoted by $\mathcal{L}(N)$. The pomset language of a Petri net is a subset of the set of all possible sequences of actions that can be performed on the net.

mm

i i i i l ll concurrent R i
i i i ll chain i ll
i i line i i i lly ll i i
li i i l i ll antichain i ll
cut i i i lly ll i i
i i i i l i i i
y i i i y i lly
i l i \mathbb{N}^A \mathbb{N}^B y i

$$\sum_{a \in h^{-1}(b)}$$

y \mathbb{N} ill i i i
i i l i l i i
i i i i i ll i i
l i Y \mathbb{N}^U
- Y Y ll
- -Y 0 -Y
- l i ly y l \mathbb{N}
- Y Y ll

Definition 1. *Posets, causality structures, pomsets*

partially ordered set (poset) i i i l
ll e_1 predecessor e_2 e_2 successor e_1 i e_1 e_2
l e_1 i direct predecessor e_2 e_2 i direct successor
 e_1 i i i lly i e i e_1 e e e_2
causality structure over i i l κ κ κ i i
 κ κ l lli i κ κ li y 1 2
isomorphic i i ij i κ_1 κ_2 i κ_2 -1 κ_1
 κ_2 κ_1 1 pomset (partially ordered multiset) i i i l
li y y i i li y \square

i lly ill i
l ll y l l i i
- i l i l
li i i i
i i i

1 \circ $) \in \circ \iff \exists$ $) \in \wedge$ $) \in$

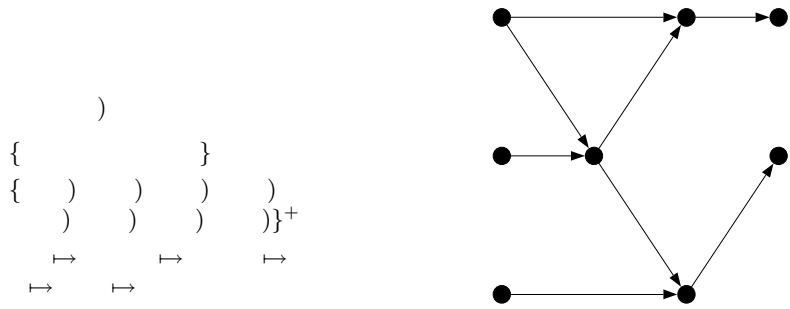


Fig. 1. $l \quad i \quad l \quad i$

$i \quad ill \quad i \quad l \quad ll \quad l \quad i \quad i$
 $i \quad i \quad ill \quad i \quad ly \quad ll \quad l \quad i \quad i$
 $i \quad i \quad y \quad lly \quad i \quad j \quad i \quad i \quad VAL$
 $l \quad MOD \quad i \quad i \quad l \quad lli$
 $l \quad y \quad l \quad l \quad l \quad i \quad i \quad l$
 $i \quad ly$

Definition 2. *Petri net*

$l \quad ll \quad i \quad net \quad i \quad il \quad i$
 $i \quad i \quad i$
 $VAL \quad \mathbb{N}^{VAL} \quad MOD \quad \mathbb{N}^{MOD} \quad VAL \quad MOD$
 $- \quad VAL \quad MOD$
 $- \quad \mathbb{N}^{v(x) \quad u(y)}$
 $i \quad ll \quad simple \quad i \quad ll$
 $l \quad i \quad i \quad labelling \quad i \quad i \quad marking \quad i$
 $l \quad \mathbb{N}^S \quad VAL \quad VAL \quad >0 \quad i$
 $i \quad marked \quad i \quad i \quad i \quad l \quad i \quad 0 \quad i \quad l \quad i$
 $l \quad i \quad f \quad \mathbb{N}^S \quad VAL \quad i \quad i \quad y \quad l$
 $il \quad i \quad y \quad 0 \quad f \quad i \quad l \quad ll \quad i \quad l \quad ll \quad i \quad i \quad l \quad l$
 $i \quad i \quad ll \quad j \quad coloured$
 $net \quad \square$
 $i \quad i \quad i \quad l \quad i \quad i \quad place/transition \ nets \quad PT-nets$
 $i \quad l \quad i \quad VAL \quad \blacksquare \quad MOD \quad y$

mm

■ y y y l i
j i y i i i l i 0
ll l l ll ily
i l i i i l i l i y
i i l i ll i *activates*
i i i y l i ²
l i *occurrence* i i y l
i — i
i *reachable* 0 i i 0
ll *interleaving behaviour* 0
i *-safe* >0 i i y i l 0 ll l
i i i l i l i i i
k l ll i
ll *T-restricted* i y i i l
l l l lly ³
li i i i y i il
ll l i i l i
i fl l i l i
i ly y

Definition 3. *Occurrence net*

i *occurrence net* *O-net* i i l i yi
ll i i i
ll i i l
l i i y i l y y ll i
l i i y li i + i i fl i
l ll *conditions events* i ly l
i □
i l i lly yi l i i i l i +
1 2 i 1 1 2 2 1 2 + 1 2 *F+*

2)) m m
m) m m
3)) m))) ∈ ∈
m m m

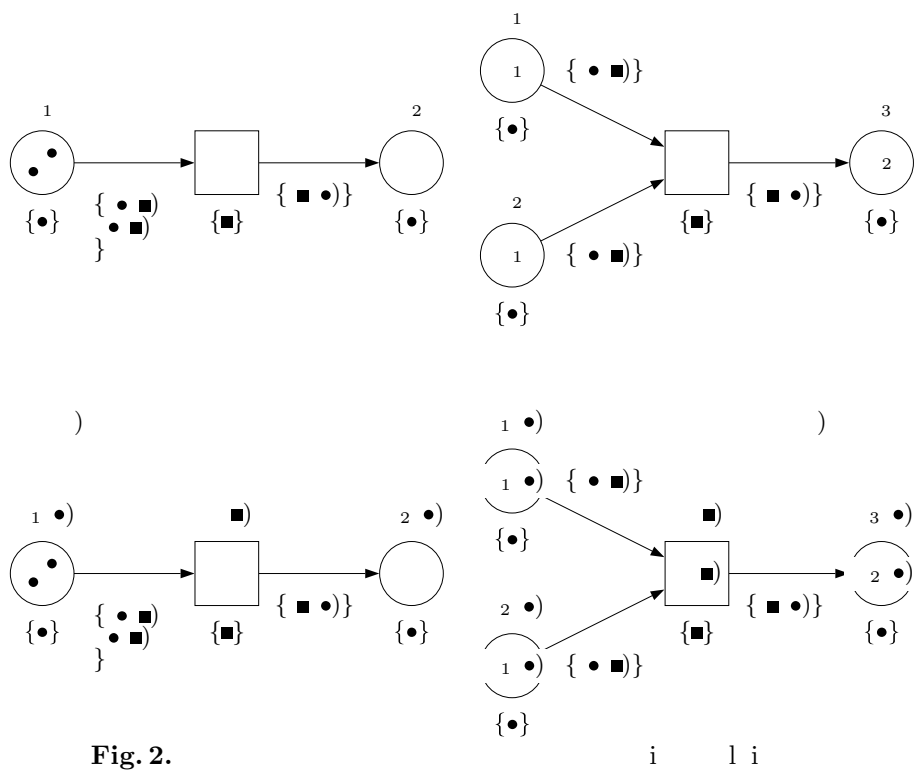


Fig. 2.

Definition 6. Pomset language

Let \mathcal{P} be a Petri net. The pomset language of \mathcal{P} , denoted by $\text{Pomset}(\mathcal{P})$, is the set of all pomsets (M, \prec) such that M is a marking of \mathcal{P} and \prec is a partial order on the events of M satisfying the following conditions:

- 1. If e and f are events of M such that $e \prec f$, then e and f are causally related in \mathcal{P} .
- 2. If e and f are events of M such that e and f are concurrent in \mathcal{P} , then e and f are incomparable in \prec .
- 3. If e and f are events of M such that e and f are concurrent in \mathcal{P} , then e and f are incomparable in \prec .

3 Balanced Coloured Nets

Let \mathcal{P} be a Petri net. The balanced coloured net of \mathcal{P} , denoted by $\text{BalCol}(\mathcal{P})$, is the Petri net obtained from \mathcal{P} by replacing each place p by a set of places $\{p_i\}_{i \in \mathbb{N}}$ and each transition t by a set of transitions $\{t_i\}_{i \in \mathbb{N}}$ such that:

- 1. The initial marking of $\text{BalCol}(\mathcal{P})$ is (M, \prec) where M is the initial marking of \mathcal{P} and \prec is the initial partial order of \mathcal{P} .
- 2. The transitions of $\text{BalCol}(\mathcal{P})$ are balanced, i.e., for each transition t_i , the number of tokens of each colour is the same.

unf i i i i i l
 i i i ll i ly i i l i i
 ll l i y i i i i i l
 l unf i i i
 unf unf
 l i i i ly i i l
 l i i i y l i l
 i i i y i y l i i i
 unf i i unf i ■
 ill ll i ll i

Corollary 1.

For any coloured net , unf . ■

Corollary 2.

For any coloured net , is -safe iff unf is -safe. In particular, for a simple with only balanced modes and a spread initial marking $_0$, unf is -safe.

Proof: l i ll i l ■
 i l l i i l l
 i l l i l ll
 l i l $basic$ i ll y i
 i l i i i y l
 l i i i y l i
 PT i i y l i ll l y
 i l ll y ■ i y i l i i ly
 i l i i i ly i i PT i ill lly
 i i l i i y ll
 y i i

Lemma 3.

PT i i i ly i PT i
 y i i

Proof: i il ■

Lemma 4.

ll li l i colour

Proof: y colour i PT colour l ly i

i colour i i l PT i i ily

l i PT l i i fl y

i i

ll colour ■

i l i l l i lly i i

i i ily i i i l colour

i i i i y l 1 2 i i l

i i i i 1 i

i 2 l ly i i i i l i

lly i j i y colour i l

l j l i

i i

i i i i \mathcal{F} i i i i

ill \mathcal{F} l ill i ly i

i i l y i l i i y i i

i i lly i l

4 A Pumping Lemma for Processes

i l ily i \mathcal{F} l ll

y ii i l li

i i ii i

ii ii j l i i i

i ily j l ly y l \mathcal{F} i

ii j l i i i l i

y li i

i l ii i y il

l i ll i i ll

l i i i i l i i l

i l ly i i l i l i l ll li

li i

Definition 9. *Line system*

li + ll i i line system i i i l

ll li y \mathcal{L} □

mm

i y li y ly l

i i i i li y l

\mathcal{F} li i i i i i i

i li y li i i y y i l i l

i i y i i li li l y y

i i l i i ly i i i l i

i i i l i l i i i

i i i li li ly i i i ill

i i j ll i li l i i

i ll

l i i y ill i i j

ly i i l i ll i l 6

l i l ly i i i

l i l ill l i

i l i i ily l y l

i y i l y

i ily l l ill l i

l i i y i \mathcal{F}

l i i i l i l i

i i i ly l

i o f li y i li

i y

Theorem 1. DILWORTH 1950

In a partial order, the maximal cardinality of a cut is equal to the minimal number of chains into which the partial order can be partitioned. ■

Lemma 5. *Maximal Number of Lines*

o f

i li y i

Proof: i i

i ly i l y il

i i l i i i l

$+_{B \ B}$ i i i i l l

i li i ily li i ll i i

6 m m m m

m

i l ly i li i li y i
li ■

li y li i i l y li y i
ll i i i i li l l i li i
li i i i li i y i
i i i i i +_{B B} i l i i l
i i i i

il i i i i li
i i y i i i iz
i y i

l i i li y i i l l i i i
i i y i i i
l i i i iz

Definition 10. *Equivalence of cuts*

\mathcal{L} $\overset{0}{\text{li}}$ $\overset{f}{y}$
 $\overset{1}{1}$ $\overset{2}{2}$ $\overset{1}{i}$ $\overset{2}{l}$ $\overset{1}{i}$ $\overset{2}{1}$ $\overset{2}{L}$ $\overset{2}{2}$ $\overset{1}{i}$ $\overset{2}{l}$ $\overset{1}{i}$ $\overset{2}{ll}$ $\overset{1}{1}$ $\overset{2}{2}$
y
 $\overset{1}{1}$ $\overset{2}{2}$ $\overset{1}{1}$ $\overset{2}{1}$ $\overset{2}{2}$ $\overset{2}{2}$

i i ij i □ 0

i y $\overset{L}{i}$ i i i l l i l ly i i $\overset{L}{L}$
y i i l yi i yi i i $\overset{L}{L}$ yi l
i l i i
i ll *repeatable* l i i li y
li i i i i l $\overset{1}{1}$ $\overset{2}{2}$ $\overset{1}{1}$ $\overset{2}{2}$ $\overset{1}{1}$ $\overset{2}{L}$ $\overset{2}{2}$
 $\overset{2}{2}$ i l i ly $\overset{1}{1}$ $\overset{2}{2}$ $\overset{1}{1}$
 $\overset{1}{1}$ $\overset{3}{3}$ yi l $\overset{1}{1}$ $\overset{3}{3}$ i i ll
l $\overset{1}{1}$ $\overset{2}{2}$ i i $\overset{1}{1}$ $\overset{3}{3}$
ll i l i i l
i i iz

$$\begin{array}{c} - \\ - \\ - \end{array} \begin{array}{c} i \\ \\ l \end{array} \quad \mathbb{N}^0 \quad n \quad Proc$$

Proof:

$$\begin{array}{c} 0 \quad f \quad 0 \quad 2^k |S| \\ i > 0 \quad l \quad y \quad l \quad i \\ i +_E E \quad i \quad li \quad i \quad i \\ i \quad y \quad lli \quad e_1 \quad e_m \\ 0 \quad 1 \quad m \quad i \quad 0 \quad i \quad i+1 \quad i \quad e_{i+1} \quad e_{i+1} \\ l \quad ly \quad i \quad i+1 \quad 0 \quad y \quad l \quad i \quad li \quad y \\ \mathcal{L} \quad i \quad li \quad y \quad l \quad 0 \quad i \\ i \quad l \quad i \\ i \quad j \quad i \quad L \quad j \quad - \quad i \quad j \quad i \quad l \quad i \\ i \quad y \quad li \quad i \quad j \quad i \quad i \quad n \quad i \quad ll \quad i \\ l \quad 0 \quad i \quad li \quad y \quad j \quad - \quad i \quad l \quad ll \quad i \\ 0 \quad i \quad j+1 \quad m \quad i \quad i \quad i \quad i \quad l \quad l \\ i \quad 0 \quad y \quad l \quad i \quad l \\ y \\ i \quad l \\ ll \quad l \quad i \quad l \quad i \quad j \quad i \quad i \quad li \quad i \\ i \quad y \quad i \quad l \quad l \quad l \quad i \quad li \quad i \quad i \quad l \quad l \\ i \quad i \quad ly \quad l \quad l \end{array}$$

5 The Equivalence of k -Safe and 1-Safe Nets

$$\begin{array}{c} i \quad i \quad l \quad y \\ i \quad l \quad i \quad l \quad l \quad i \quad i \quad i \quad l \\ l \quad i \end{array}$$

Lemma 8.

$$\begin{array}{c} ll \\ i \quad ll \end{array} \quad i \quad l \quad i \quad colour \quad colour$$

Proof:

$$\begin{array}{c} ly \quad i \quad y \quad l \quad i \quad i \\ 0 \quad f \quad 0 \quad \mathbb{N} \\ i \quad l \quad 1 \quad m \quad Proc \quad 0 \quad i \\ l \quad i \quad i \quad i \end{array}$$

$$\mathrm{mm}$$

$$\begin{array}{cccccccccccccccc} i & & i & & l & & & & Proc & & 0 & & & & i & & 2 \\ & & i\ i & & & & & & & & i & & 1 & & i & & \\ i & & m & & min & & l & & l & & i & & l & & lli & & i \\ & & & & & & & & & & & & & & & & min \\ & & & & & & & & l & & \Psi_{i,j} & & i & & i & & j\ i \\ & & & & & & & & & & & & i & & \Psi_{i,j} & & \end{array}$$

$$\mathcal{F} \bigcup_{1 \leq i \leq m, 1 \leq j \leq k \mid S|} \Psi_{i,j}$$

$$\begin{array}{cccccccccccccccccccc} l & & i & & i & & li & & y & & \pi_i & & \mathcal{L} & & i & & i & & \pi_i & & i \\ & & & & & & li & & y & & ij & & i & & \alpha_{\pi_i} & & \pi_i & & & & \pi_i \end{array}$$

$$\begin{array}{cccccccccccccccc} min & & \mathcal{F} & & & & y & & \bigcup_1 & & i & & m,\ell & & L_{\pi_i},b & & \ell & & \Psi_{i,\alpha_{\pi_i}(\ell)} \\ & & 0 & & f & & \mathbb{N}^{S-2} & & y & & 0 & & 0 & & 0 & & 0 & & min \end{array}$$

$$\begin{array}{cccccccccccccccccccc} \Psi_{i,j} & & i & & 0\ i & & & & i & & & & & & & & & & & & \\ & & lly & & i\ j\ i & & 0 & & discolour & & 0 & & y & & i\ i\ i\ li\ y & & i\ i \\ & & i\ i & & ll & & & & i\ i & & i & & i & & ly & & y \\ & & i\ y & & 0 & & i\ y & & colour & & i & & ll & & & & & & & & \\ & & i & & & & & & i\ i\ i & & y & & i & & & & & & & & \\ i & & i & & & & li & & i & & y & & i & & & & & & & & \\ & & & & & & colour & & i & & & & & & & & & & & \end{array}$$

$$\begin{array}{cccccccccccccccccccc} & & \pi & & \pi & & Proc & & & & i & & & & & & & & & & \\ & & 0 & & l & & ly & & i & & i & & i & & i & & & & & & \\ & & ill & & l & & lli & & i & & l\ i & & j & & i & & & & & & \\ colour & & i & & & & i & & i & & i & & i & & i & & l & & & & \\ i & & \mathcal{F} & & i & & y & & i\ i\ i & & min & & ll & & i\ i & & i \\ i\ i\ lly & & y & & i\ i\ i & & min\ i & & i & & i & & l & & i & & i & & i \\ \Psi_{k,j} & & i\ l\ i & & ill & & li & & \Psi_{k,j} & & l & & ly & & i & & & & & & \\ i\ i\ i & & i & & i & & i & & & & li & & & & \pi_i & & & & & & \\ & & i\ i & & & & y & & \Psi_{i,\alpha_{\pi_i}(\ell)} & & i\ ly\ i\ e\ i & & & & & & & & & \\ i & & li & & \pi_i & & i & & i & & e & & l & & i\ l\ y & & & & & & \\ & & y & & i & & \Psi_{i,\alpha_{\pi_i}(\ell)} & & l & & i & & i & & y & & & & & & \\ i\ i & & e & & i & & l\ li & & \Psi_{i,\alpha_{\pi_i}(\ell)} & & i & & li & & y & & \pi_i & & & & \\ & & l & & & & ll & & i & & i & & i & & \Psi_{k,j} & & & & & & \\ ll & & i\ i & & i & & i & & i & & i & & i & & & & & & & & \\ i & & l & & li & & i & & i & & & & ll & & & & & & & & \\ & & \Psi_{k,j} & & i & & l & & l & & \bigcup_b & & F(.,e) & & \bigcup_b & & F(e,.) & & & & \\ li & & i & & i & & l & & l & & i & & y & & & & & & & & \end{array}$$

m

y i l i ll $colour$ l y e i y i j
 y i l i ll $colour$ l ll i i i i l y li y
 y i l i ll i $colour$ i ily li y
 ly i i lly l i i $\Psi_{i,j}$ 8 l l
 $>$ 0 li i y i l ly
 l i i l lly i l li i
 y i li y i l i l
 i i li i ll li l i
 l i y i l y li
 i i i l i i i ly i i ll
 i ll $colour$

■

i i l y ill
 i i i $\Psi_{i,j}$ i i
 ll \mathcal{F} l l l i i i 0
 0 i

0	$\Psi_{1,1}$	$\Psi_{2,1}$	$\Psi_{3,1}$	$\Psi_{4,1}$	1
0	$\Psi_{1,2}$	$\Psi_{2,2}$	$\Psi_{3,2}$	$\Psi_{4,2}$	2
0					0

y

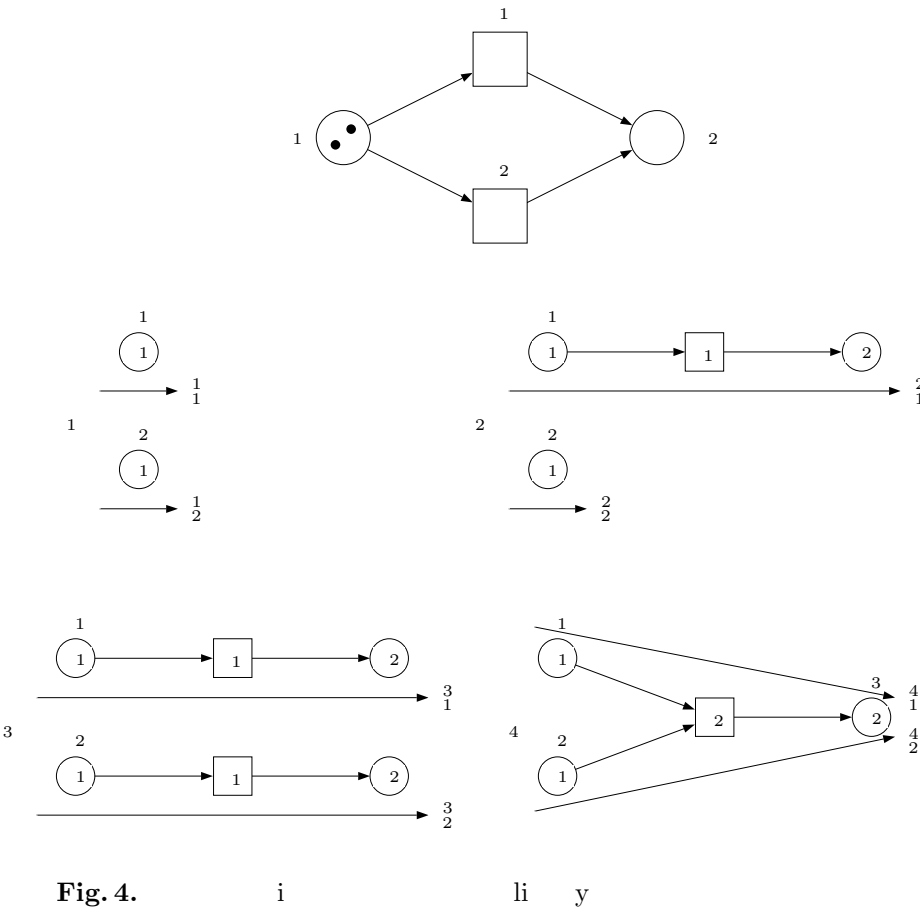
i i 4 i l i $\Psi_{1,1}$ $\Psi_{4,2}$
 ill i y i i i i 3
 i lly i i i i i

Theorem 2. MAIN RESULT

For every δ -safe PT-net \mathcal{N} there is a δ -safe PT-net \mathcal{N}' with $\mathcal{N} \sim \mathcal{N}'$.

8

$\Psi_{i,j}$



Proof:

$PT \text{ unf colour}$

$colour$

$ll \ y$

$unf \ colour$

$ll \ y$

$PT \ unf \ colour$

$i \ y \ ll \ i$

$ll \ y \ li \ colour \ i \ y \ ii \ i \ l \ ly$

$l \ unf \ colour \ i \ i \ y \ ii \ i \ ll \ i \ ly$

■

A diagram showing a sequence of points labeled with 'm' and 'mm' arranged in a roughly circular pattern. The labels are: 'mm' at the top, 'm' at the top-right, 'm' at the right, 'm' at the bottom-right, 'm' at the bottom, 'm' at the bottom-left, 'mm' at the left, 'm' at the top-left, and 'm' at the top. There are also some smaller 'm' labels scattered within the circle.

Executing Transactions in Zero-Safe Nets*

un n n n
m m
{bruni,ugo}@di.unipi.it
http://www.di.unipi.it/~bruni
http://www.di.unipi.it/~ugo/ugo.html

Abstract

m m m m
m PT m m
m
m transactions m
m zero-safe nets PT
m m m
m m
m m
m PT m
m m m
m m

Keywords: PT

Introduction

u n v ll n v l n n
n v lv n u n l n l l n u n l n n
n n n
n n n l l u n n u n n
n l l v n l n n
v PT n v ul u u
n n n l l u u n l
n n un u l n n u n
u l n n n v n n v l u
un n l l n u n u n n n l n u
u ll n n u v l n n n v l

* *Progettazione e Verifica di Sistemi*
Eterogenei Connessi mediante Reti *CONFER2* *CO-*
ORDINA *Tecniche Formali per Sistemi Software*
TOSCA: Tipi, Ordine Superiore e Concorrenza

u n l PT n l u n n
 n n n u n n u n n
 u n v l u n u n n n l u
 l l v n l k v l u u n n v
 n n
 n n n l n l v PT n nv
 n n u u l n n v l n n
 n n n n l v
 n n un n n l v
 l un n l n n v n n n nn
 l v u n n n n n n n ul
 n n n n n u kn l n
 v u n u u n nl l l l
 u l n n l n
 n n n n l v l l l u
 u n u n nl ll nv lv n n l
 k n ll n ul n
 n n n ll n n n l l k u
 n n u u n n u
 n n u u n n nl
 ll u n nl l u n n v
 l n n PT n
 n l u n n n PT n n u
 n n n n n u l k ll u u
 ll n u n u n ll n n n
 ul l k u n n n
 nl n u n l
 l kn n u n n n n un¹ n
 n n n n un n n n n u
 v n n n nl n n n n u
 n u n l kn n n n ll n
 n ll n n n l kn
 u n l n u l kn n n
 v l v n l u n n k
 n n n u n n n u n
 n n n v n n
 n n v l n n un n ll
 n l n n u l n n n u l u n
 l n n n u u ll u n l n
 n l n n n nn n
 l l n n

¹

free choice

t_1 t_2
 s

t_1 t_2

m

$\{s\}$

ll n n u n n n n v
 u u n u n n n l n n
 n
 u u n l l n n n ll
 n n n n un n
 n ul n n 1 n 2 n n 1 n 2
 u kn u n n n n
 1 n 2 u n n 1 n 2
 ZS n v n n u n v n n n
 n n n PT n n n n nl n n kn
 n n n n l ll l ZS n n lu
 l l u n nv l
 n l v l u n kn n
 l l n v l n
 v lu n ZS n l kn v lv u n
 kn kn n l n v n u ll n
 n l kn v ll l kn u u n n
 l n n u v kn
 n n n n 1 u kn n n l
 n l u n n 2 l
 n u n u n n l v 1 n
 2 l k n n l v l u u n v n
 ZS n n n PT n u l v n
 l nl l l n n n v
 n n u v n
 u n n n u n kn n v
 un u u n l v n u l
 ul n n l l k
 n n n v lu n u l
 nn v n n ul n n ZS n l l
 n l n n ZS n n
 u n l ll n v n
 n n n n ul un v l
 n u n n l n u n n
 v l l n u n n n
 u u l n n l n
 un n u n u n ZS n

Dining Philosophers and Free Choice Nets. l l
 ll u n u n llu l n ll kn n
 l n n l l n
 un l v n l n n n n
 u l l u k l k n l l

l ll nk n u n k n l n
n n l n n u ul l
u n k n l n nk n n

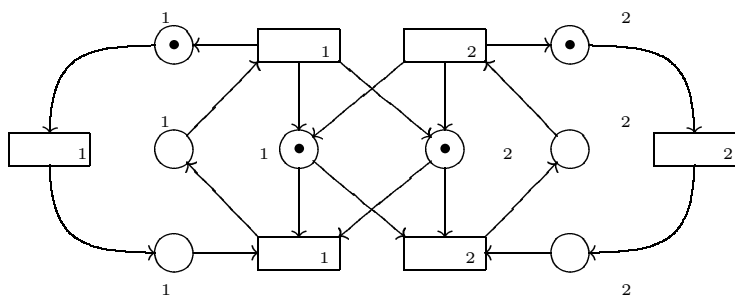


Fig.1. n v n n l

PT n ll u n u u l l
n n l n n n ul k n ull
n fl l n un
k n n n l n nk n n n
v l k n n l k n k n l k
n n k n n n n k l k
k n n n v l n n k u
k n u nn l l n
n l k n n n k n n l i
2 k₁ n k₂ l nk n n k n
l
u l n ll l
u n n k n v l n u
n n k n l v l l
u n n k ul k k
l n mod k l n n n
ul n l k l k n k n n
n n nu n nfl n n n
n n n nul
u n l l n n n l u n
n ll n l ll l k n u n
l l u n n u n n u n n
u n u ll u ll u n

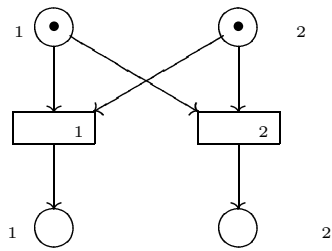


Fig.2. $n \quad l \quad n \quad n \quad n$

$n \quad n \quad n \quad l \quad n \quad n \quad n \quad n \quad n \quad n \quad l \quad n$
 $n \quad n \quad n \quad l \quad l \quad n \quad u \quad n \quad l \quad n$
 $n \quad n \quad n \quad n \quad u \quad n \quad n \quad u$

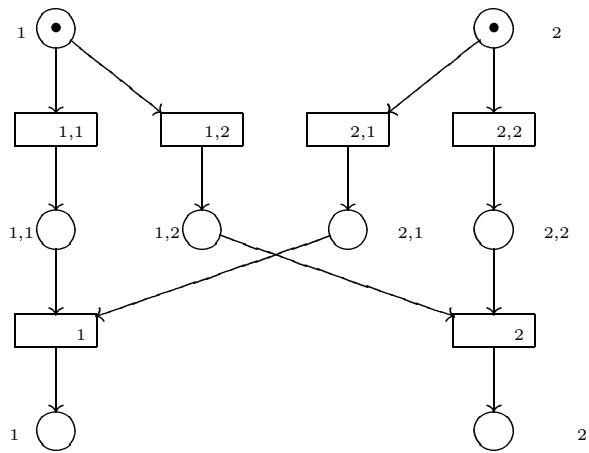


Fig.3. $l \quad n \quad n \quad n$

$k \quad l \quad n \quad n \quad n \quad l \quad n \quad n \quad n \quad n \quad n \quad n \quad n \quad k$
 $n \quad n \quad n \quad k \quad u \quad ll \quad n \quad n \quad n$
 $k \quad n \quad l \quad l \quad 1 \quad 1 \quad n \quad n \quad k \quad n$
 $n \quad u \quad u \quad n \quad l \quad 2 \quad 2 \quad n \quad n \quad u \quad n \quad n \quad ll \quad n$
 $u \quad n \quad n \quad l \quad n \quad n$

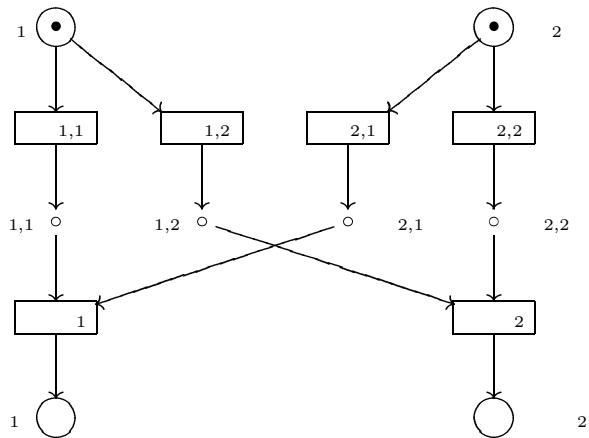


Fig.4.

n ll v n l u n nl
 n n n k n u nl l v l u l
 n u n u l nv l u n l k
 n n l n u n
 n 1 1 n 2 2 n u n n nv l
 n n k n u n l l v l l
 ZS n ll l n l
 n n n l u v n n
 n n l k n n

Concurrent Strategies, Operationally.

k ZS n n u l l
 v n l n v ll n u n
 u u n l n n nl
 n n v n l k n n n n l l v l u n
 v n n u n n ul l
 v n n l n n n l n n l
 l lu n u u n v
 n ll u u l u n n u
 n n l l n ul ll n v l u n v n
 ZS n n l k n l u n u
 n n n k n n v
 u n n n n lu n l n n
 n un l n PT n n
 ul n n n u n n n n n

in

$v \mid \bar{} \quad u \mid \bar{} \quad \parallel \quad n \quad u \quad n \quad n \quad n \quad n$

$n \quad n \quad n \quad n \quad l \quad l \quad n \quad l \quad n \quad \bar{} \quad \bar{}$

$n \quad n \quad n \quad n \quad ul \quad n \quad l \quad n \quad n \quad l \quad u$

$n \quad n \quad l \quad n \quad ll \quad l \quad n \quad v \quad n \quad u \quad n \quad u \quad n \quad n$

$n \quad l$

$\frac{u \in S^{\oplus}}{u \Rightarrow_N u}$	(identities)
$\frac{t \ u \rightarrow v \in T}{u \Rightarrow_N v}$	(generators)
$\frac{u \Rightarrow_N v, \ u' \Rightarrow_N v'}{u \oplus u' \Rightarrow_N v \oplus v'}$	(parallel composition)

Table1: $n \quad n \quad ul \quad \bar{} \quad \bar{}$

$n \quad n \quad n \quad ul \quad n \quad n \quad 0 \quad 1 \quad n \quad u$

$n \quad u \quad v \quad l \quad n \quad ll \quad u \quad n \quad v \quad n$

$u \quad n \quad n \quad n \quad nn \quad l \quad v \quad u \quad n \quad n$

$n \quad n \quad n \quad n \quad n \quad n \quad n \quad n \quad n$

$u \quad l \quad l \quad n \quad n \quad n \quad n \quad n \quad n$

$n \quad l \quad n \quad ll \quad v \quad k \quad n \quad l \quad CTph$

$n \quad ul \quad n \quad n \quad u \quad n \quad n \quad n \quad l$

$u \quad u \quad u \quad nl \quad n \quad u \quad n \quad n \quad u \quad v$

$l \quad n \quad ll \quad k$

$n \quad ll \quad u \quad v \quad l \quad n \quad u \quad v \quad n \quad n \quad n \quad n$

$n \quad n \quad n \quad n \quad l \quad n \quad v$

$k \quad n \quad n \quad v \quad u \quad l \quad k \quad n \quad l \quad ITph \quad n \quad ITph$

$u \quad l \quad n \quad n \quad n \quad l \quad n \quad n \quad v \quad lu \quad n \quad n$

$n \quad u \quad n \quad nl \quad u \quad n \quad n \quad n \quad ull \quad n \quad n$

$CTph \quad n \quad u \quad v \quad l \quad n \quad l \quad n$

$n \quad v \quad ll$

$v \quad n \quad v \quad n \quad u$

$k \quad n \quad un \quad u \quad l \quad u \quad n \quad u$

$$\begin{array}{c}
 n \qquad \qquad \qquad \text{ull} \qquad v \qquad \qquad n \qquad n \qquad l \ n \\
 n \qquad \qquad \qquad l \qquad \qquad n \qquad n \qquad n \qquad n \qquad u \ n \ l \\
 l \qquad \qquad n \qquad n \qquad n \qquad CTph \ n \ ITph \ n \ n \qquad n \\
 v \ u \ \text{ul}
 \end{array}$$

———— (one-step computations)

———— (sequential composition)

$$\begin{array}{c}
 n \qquad n \qquad u \qquad \qquad n \ v \ l \ u \qquad n \qquad \qquad n \qquad \qquad n \qquad n \\
 n \ u \ n
 \end{array}$$

1.2 Zero-Safe Nets

$$\begin{array}{c}
 n \qquad u \qquad n \ PT \ n \qquad \qquad l \ l \qquad ll \qquad \qquad l \\
 n \qquad v \qquad k \ n \ fl \qquad \qquad u \ n \qquad v \ l \ n \ n \\
 n \qquad \qquad v \qquad n \ ll \qquad \qquad n \qquad n \ n \ ll \qquad \qquad n \\
 n \ n \qquad \qquad v \ n \ n \ n \qquad \qquad n \qquad n \qquad n \qquad n \\
 n \ n \qquad \qquad n \ n \qquad \qquad n \ n \qquad \qquad n \ l \qquad \qquad n \\
 v \ l \ ul
 \end{array}$$

Definition 1.2 (Zero-safe net).

$$\begin{array}{c}
 n \qquad \qquad \qquad ZS \\
 \text{in} \qquad \qquad \qquad \text{in} \\
 un \ l \ n \ PT \qquad \qquad \qquad l \qquad \qquad l \qquad k \ n \\
 \qquad \qquad \qquad \text{in} \\
 l \qquad k \ n \qquad \qquad v \ l \\
 n \qquad n \qquad \qquad k \ n \ n \qquad v \ n \qquad k \ n \ n \qquad \qquad n \ n \ l \ n \\
 n \ un \qquad v \ l \qquad ll \qquad \qquad n \qquad \qquad k \ n \\
 v \ l \ l \ n \qquad \qquad l \ l \qquad n \qquad \qquad l \\
 n \qquad \qquad \qquad n \qquad \qquad u \ v \ n \\
 n \qquad u \qquad \qquad \times \qquad n \qquad \qquad n \qquad \qquad n \qquad n \\
 n \qquad \qquad \qquad n \qquad \qquad \qquad \qquad \qquad \qquad \qquad l \ ul \\
 n \qquad n \qquad \qquad ul \qquad v \\
 PT \ n \qquad \qquad n \ n \qquad \qquad v \qquad ZS \ n \qquad \qquad n \\
 l \ n \ - \Rightarrow \ - \qquad n \qquad \qquad n \ n \ ul \ n \qquad l \qquad n \ u \ l \qquad l \ n \\
 - \Rightarrow \ - \qquad n \ u \qquad \qquad l \ n \\
 \qquad \qquad \qquad n \ k \ v \ n \qquad \qquad l \ n \ - \qquad \qquad \qquad \text{underlying} \\
 l \ n \ n \qquad \qquad n \ u \ n \ l \qquad u \ n \qquad v \ l \ n \qquad n \ ul \qquad \qquad \qquad \text{un} \\
 ul \ \text{horizontal composition} \qquad \qquad ll \ l \qquad \qquad n \qquad \qquad l \ u \\
 n \qquad \qquad u \ n \ l \qquad \qquad n \qquad \qquad l \qquad \qquad ll \qquad \qquad n \ l \qquad u
 \end{array}$$

$\frac{u \oplus x \Rightarrow_{NB} v \oplus y, \quad u, v \in (S \setminus Z)^\oplus, \quad x, y \in Z^\oplus}{(u, x \rightrightarrows_B (v, y$	(underlying)
$\frac{(u, x \rightrightarrows_B (v, y, \quad (u', y \rightrightarrows_B (v', y')}{(u \oplus u', x \rightrightarrows_B (v \oplus v', y'}$	(horizontal composition)
$\frac{(u, 0 \rightrightarrows_B (v, 0}{u \Rrightarrow_B v$	(commit)

Table2:
n n ul - \Rrightarrow -

$$\begin{array}{c}
\begin{array}{ccccccc}
& & v & & n & n & n & & n & & n \\
n & & u & n & l & & n & & u & n & & fl & v & & ll \\
& & n & & ul & \textbf{commit} & l & & n & n & & n & & n \\
& & l & & u & & l & & kn & n & n & u & n & & l \\
& & kn & & & & ul & n & n & & n & l & & n \\
n & & n & & n & & \Rightarrow & & n & & \Rightarrow & & n & v \\
ll & l & & n & & \Rightarrow & & & & & & & &
\end{array}
\end{array}$$

Abstract Net.
n
l v l
l v
ZS n
n
u v l n l
v
PT n
 \mathcal{N}
u
()
n
-
() -
- \Rrightarrow -
n
v
l PT n
v
n
n
n
l n
l n
n
ll
n
n
u
n
u
n
n
n
 \Rightarrow
kn u
u v l n
u
n
n u
n
v n
n
nn
n
n
n
n
n
v
n u
n
kn l
v
n
n
n
ll
n
n
kn
n
u
n
v
n
n
n
n
n
l n u
n
un
n
n
n
u
n
n
u
v
v
l n
n
n
l
n
ZS n
v
n
u
n
n
n
fl
n
un v
l
n
l u
n
u
n
n
un
u
u
n

$ITph$
n
v
l
n
u
n
n
u
u
n
 $CTph$
l
n
u
n
l
n
u
n
v
n
v

2 An Operational Definition for Transactions

$$\begin{array}{c}
 u \quad n \quad n \quad n \quad n \quad ll \quad n \quad n \\
 l \quad n \quad ZS \quad n \quad n \quad v \quad l \quad n \quad n \quad l \quad n \\
 n \quad ul \quad n \quad v \quad l \quad u \quad l \quad n \quad l \\
 n \quad l \quad n \quad n \quad n \quad n \quad n \quad u \quad n \\
 un \quad l \quad n \quad n \quad ul \quad n \quad n \quad n \quad n \quad n \quad un \\
 v \quad n \quad n \quad n \quad n \quad l \quad n \\
 n \quad u \quad n \quad n \quad k \quad n \quad u \quad l \quad n \\
 k \quad kn \quad n \quad l \quad n \quad n \quad n \quad l \\
 n \quad n \quad l \quad n \quad n \quad n \quad lu \quad n \quad nn \quad n \\
 u \quad n \quad n \quad ZS \quad n \quad n \quad u \quad n \quad v \\
 l \quad v \quad l \quad u \quad l \quad u \quad n \quad un \quad l \quad n \quad n \quad ul \quad n \\
 n \quad n \quad u \quad n \quad nn \quad n \quad n \quad u \\
 n \quad n \quad n \quad v \quad n \quad n \quad u \\
 u \quad n \quad l \quad l \quad n \quad un \quad l \quad n \quad u \\
 v \quad n \quad u \quad u \quad l \quad n \quad v \quad n \\
 u \quad u \quad n \quad u \quad nl \quad ll \quad n \quad n \quad n \quad un \quad ZS \quad n \quad un \quad n \quad n \\
 n \quad u \quad n \quad n \quad n \quad n \quad u \quad n \quad n \quad n \quad n \quad u \quad n \\
 u \quad n \quad n \quad l \quad n \quad - \Rightarrow \quad - \quad n \quad n \quad u \\
 l \quad ll \quad n \quad l \quad u \quad n \quad kn \\
 n \quad n \quad n \quad n \quad n \quad \mathcal{R} \Rightarrow \\
 v \quad lu \quad n \quad n \quad l \quad u \\
 n
 \end{array}$$

2.1 PT Net Unfolding

$$\begin{array}{c}
 un \quad l \quad n \quad n \quad v \quad n \quad u \quad v \quad n \quad ll \quad l \\
 u \quad n \quad n \quad l \quad n \quad n \quad l \\
 n \quad n \quad n \quad n \quad n \quad u \quad l \quad n \quad n \quad u \quad n \quad n \quad un \quad l \quad n \\
 n \quad u \quad n \quad ll \quad n \quad n \quad PT \quad n \quad n \\
 l \quad n \quad ul \quad n \quad u \quad n \quad n \quad v \quad u \quad n \\
 u \quad u \quad l \quad n \quad n \quad v \quad v \\
 kn \quad u \quad n \quad n \quad l \quad l \quad n \quad u \quad ul \quad n \quad n \\
 n \quad n \quad l \quad n \quad n \quad u \quad l \quad n \quad n \quad ul \\
 n \quad kn \quad l \quad n \quad n \quad n \\
 n \quad u \quad n \quad v \quad u \quad n \quad PT \quad n \quad k \\
 un \quad l \quad n \quad l \quad nl \quad k \quad n \quad u \quad n \quad n \quad l \quad kn \\
 n \quad n \quad un \quad l \quad n \quad u \quad n \quad n \quad n \quad u \\
 n \quad n \quad \mathcal{U} \quad n \quad l \quad n \quad n \quad n \quad kn \\
 n \quad ul \quad n \quad l \quad n \quad n \quad n \quad \mathcal{U} \\
 k \quad nfl \quad n \quad ll \quad n \quad un \quad l \quad n \quad n \quad n \\
 ll \quad l \quad n \quad n \quad n \quad un \quad l \quad n \quad n \quad n
 \end{array}$$

l n n l \mathcal{U} n k n n n n
 ll u n n n ll l un
 n k n n n fl n
 l n n l n n
 n l n $-$ $-$ $-$ n **co** $-$ $-$ v l ll l n
 $-$ $-$ n v n fl v l u l n
 $-$ 0 $-$ n

0 ^{def}
 l n nfl l n n n l l n
 n n $-$ 0 $-$ n l n $-$ $-$
 u 1 2 n 1 2 n 1 n l 2
 1 0 2 ^{def} 1 2 1 2
 n nfl l n u fl v n $-$ $-$ n $-$ $-$ v
 n n n nu n l n n l n **co** 1 2
 n 1 2 2 1 1 2 n ul l n
co u u ll n n l n n **co** n nl
 ll 1 2 v **co** 1 2
 n l \mathcal{U} v
 v n u l nu u n u n k n
 n l un n n n
 n u n v n n u k n
 k n n l k n n l ul n n n \mathcal{U}
 n n n n l l n
 n n n u k n
 nn n n n n ll
 n \mathcal{U} n n l n n ul n l

$\frac{\text{in}(\) = 1}{S_{\mathcal{U}(N)}}$									
$\frac{\begin{array}{l} : \quad \bigoplus_{j \in J} j \ j \quad N \quad = \quad i \ i \ i \quad S_{\mathcal{U}(N)} \quad \text{co}(\) \quad = \quad \bigoplus_{i \in I} i \end{array}}{\begin{array}{l} = \quad \mathcal{U}(N) \quad = \quad j \quad 1 \quad j \quad S_{\mathcal{U}(N)} \quad \text{pre}(\) = \quad \text{post}(\) = \end{array}}$									

Table3: un l n \mathcal{U}

n v u n l n n u ul ul
 n n l k n \mathcal{U} n ul un l n
 l k nu n k n n l n n ll
 l k v n n u n n u
 k n n n **co** n lu v l

n n nn l u v n
 n n l n n ll n
 l l n n kn n
 n u v n l n n nu n
 n n nu un l kn vn n
 l n u n n u vl v
 v l u n n ul n l nu nl n
 un l n n l n u l

2.2 ZS Net Unfolding

un l n un l n n n l ul n n
 v n u nu n l u
 n n n n l kn
 u n ZS n n u l l
 n l n n n 1 n 2
 — 1 n 1
 — 2 n 2
 n in n nn n n u l u
 1 n l n u n n 2 u u n n n
 nu n kn n u ul u l nu n
 kn n n n 1 n vl l
 n \mathcal{R} n n l kn n
 nu kn n u l n n n 2 1 n
 \mathcal{R}

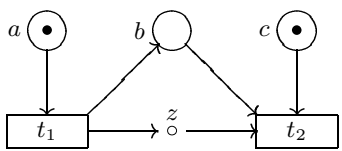


Fig.5. ZS n l

u n un l n n l nl v
 u l l n n n nu ll
 kn u
 n \mathcal{U} n l n n ul n
 l un l n n u l kn
 \mathcal{R} n l kn

$\frac{(\quad)=1}{S_{\mathcal{U}(B)}}$	
$\frac{\begin{array}{c} :(\quad) \quad (\bigoplus_{j\in J} j\ j) \quad B \\ = \quad \mathcal{U}(B) \quad = \quad j \end{array}}{\mathcal{R}(\quad)}$	$\frac{\begin{array}{c} = \quad i \quad i \quad i \quad S_{\mathcal{U}(B)} \quad \mathbf{co}(\quad) \\ 1 \quad j \quad S_{\mathcal{U}(B)} \quad \mathbf{pre}(\quad) = \quad \mathbf{post}(\quad) = \end{array}}{\begin{array}{c} \mathbf{ZProd}(\Gamma) = \mathbf{ZCons}(\Gamma) \\ \ominus \mathbf{SCons}(\Gamma) \quad \mathbf{SProd}(\Gamma) \quad \mathcal{R}(\quad) \end{array}}$

Table4:
un l n \mathcal{U}

ul	n	un l n	n	l	l
l	n	n	l	l	k n
n u	k n	v	l	n	l
	n	n	\oplus	k n	u
u	n	l		k n	n
	l	n n	\oplus	n n	l
n n	k n	ll n	l	un l	l
ll	u				
ul	v u	u	ul	n	n
n	n	n	n	u	ll
n v n	u n	k n	v l	n u	n
n	n	l	l		u

$\mathbf{ZCons} \stackrel{\text{def}}{=} \bigcup$

$\mathbf{ZProd} \stackrel{\text{def}}{=} \bigcup$

\mathbf{ZCons}	l l	\mathbf{ZProd}	k n	v	n	n u
u		n	n	k n	v	n
l	\mathbf{ZProd}	n l	n	k n	n u	n
u		k	\mathbf{ZCons}	\mathbf{ZProd}	u	k n
l	l	n	n	n	k n	
l	l	u	n	n	u	n
n	k n	v	n	n u	n	ll
n	n	u			u	l

$$\begin{array}{l} \mathbf{SCons} \stackrel{\text{def}}{=} \bigoplus \\ : (\quad) \quad (\quad) \\ \mathbf{SProd} \stackrel{\text{def}}{=} \bigoplus \\ : (\quad) \quad (\quad) \end{array}$$

$$\begin{array}{c} u \quad un \quad n \quad v \quad n \quad n \quad v \quad n \quad n \\ v \quad u \\ u \quad ul \quad k \quad \Gamma \quad n \quad u \quad n \quad v \quad n \quad n \quad k \\ k \quad n \quad u \quad v \quad n \quad n \quad u \quad n \quad n \quad n \\ v \quad n \quad n \quad \Gamma \quad l \quad n \quad n \quad n \quad nv \quad n \quad n \quad l \quad u \quad l \\ \mathbf{ZProd} \quad \Gamma \quad \mathbf{ZCons} \quad \Gamma \quad n \quad n \quad k \quad n \quad n \quad \mathbf{ZProd} \quad \Gamma \quad n \\ n \quad n \quad l \quad n \quad v \quad n \quad \Gamma \quad n \quad un \quad u^2 \\ u \quad n \quad ul \quad n \\ \mathcal{R} \quad ul \quad n \quad u \quad n \quad l \quad u \\ n \quad u \quad ll \quad n \quad n \quad v \quad n \quad n \quad \Gamma \quad u \quad n \quad ul \quad v \\ n \quad u \quad u \quad n \quad ul \quad n \quad n \quad n \quad n \\ l \quad l \quad ll \quad n \quad ul \end{array}$$

$$\text{Proposition 2.1.} \quad \Gamma \quad (\quad) \quad \text{co} \quad \Gamma \quad \mathbf{ZProd} \quad \Gamma \quad \mathbf{ZCons} \quad \Gamma$$

$$\begin{array}{c} n \quad n \quad u \quad u \\ l \quad k \quad n \quad n \quad k \quad n \quad n \quad n \\ \Gamma \quad (\quad) \quad u \quad v \quad n \quad n \quad n \quad ul \quad n \quad l \quad n \\ k \quad n \quad n \quad n \quad \mathbf{ZProd} \quad \Gamma \quad n \quad \mathbf{ZProd} \quad \Gamma \\ \mathbf{ZCons} \quad \Gamma \quad ll \quad n \quad v \quad n \quad n \quad \Gamma \quad n \\ n \quad v \quad n \quad u \quad u \quad n \quad v \\ \Gamma \quad n \quad \text{co} \quad \Gamma \quad n \quad lu \quad n \\ ul \quad n \quad l \quad u \quad u \quad l \quad l \quad n \quad n \\ l \quad n \quad n \quad ll \quad u \quad ul \quad l \quad n \quad n \\ u \quad l \quad k \quad n \quad l \quad n \quad n \quad un \\ u \quad n \quad u \quad l \quad v \\ n \quad \mathbf{ZProd} \quad \mathbf{ZCons} \quad \mathbf{SProd} \quad n \quad \mathbf{SCons} \quad ul \quad l \\ n \quad n \quad l \quad l \quad u \quad n \quad l \quad l \quad ul \\ n \quad n \quad n \quad ul \quad n \quad ul \quad ll \quad n \end{array}$$

$$\text{Theorem 2.1.} \quad \mathcal{R} \quad \Rightarrow$$

$$\begin{array}{c} n \quad lu \quad n \quad \mathcal{R} \quad \Rightarrow \quad n \quad n \quad v \quad ul \\ n \quad u \quad n \quad n \quad \mathcal{R} \quad n \quad n \quad n \quad l \quad ul \\ \text{identities underlying } n \text{ commit} \quad n \quad lu \quad \Rightarrow \\ \ominus \mathbf{SCons} \quad \Gamma \quad \mathbf{SProd} \quad \Gamma \quad \Gamma \quad (\quad) \quad u \quad \text{co} \quad \Gamma \quad n \\ \mathbf{ZProd} \quad \Gamma \quad \mathbf{ZCons} \quad \Gamma \quad n \quad n \quad u \quad n \quad u \quad n \end{array}$$

$$\begin{array}{c}
\begin{array}{c} \Gamma \quad l \\ n \ u \quad ll \\ l \quad l \end{array} \quad \begin{array}{c} k \ n \\ v \ n \quad n \end{array} \quad \begin{array}{c} k \ n \quad \mathbf{SCons} \ \Gamma \quad n \ lu \quad n \quad n \\ u \quad v \quad u \quad u \quad n \end{array} \\
\\
\Gamma \\
\begin{array}{c} u \quad n \ u \quad ll \quad l \quad k \ n \quad u \quad n \quad n \\ v \ n \quad n \quad n \quad l \quad \mathbf{SProd} \ \Gamma \quad v \end{array} \\
\\
\mathbf{SCons} \ \Gamma \quad \Rightarrow \quad \mathbf{SProd} \ \Gamma \\
\\
\text{ul identities } n \text{ underlying } v \ l \\
\\
\ominus \mathbf{SCons} \ \Gamma \quad \Rightarrow \quad \ominus \mathbf{SCons} \ \Gamma \\
\\
n \ ll \quad n \\
\\
\Rightarrow \quad \ominus \mathbf{SCons} \ \Gamma \quad \mathbf{SProd} \ \Gamma \\
\\
ll \ n \quad \ominus \mathbf{SCons} \ \Gamma \quad \mathbf{SProd} \ \Gamma \quad n \quad l \ n \quad \text{ul commit} \\
\\
\begin{array}{c} n \Rightarrow \\ nv \quad l \end{array} \quad \begin{array}{c} v \quad n \ lu \quad n \\ v \end{array} \quad \Rightarrow \quad \mathcal{R} \quad \begin{array}{c} n \ u \quad n \\ n \end{array} \\
\\
n \quad \Rightarrow \quad \begin{array}{c} k \ n \quad n \quad n \quad un \ l \ n \quad n \end{array}
\end{array}$$

3 Related Issues

3.1 Computing the Abstract Net

$$\begin{array}{c}
\begin{array}{c} n \quad un \ l \ n \quad n \\ n \ n \quad l \quad u \end{array} \quad \begin{array}{c} n \quad n \quad v \\ u \quad n \end{array} \\
\\
n \quad v \quad l \quad v \quad n \quad n \quad l \\
\\
u \quad l \quad n \quad l \quad n \quad v \quad n \\
\\
\begin{array}{c} n \quad u \quad n \quad n \quad n \quad l \quad k \ n \quad n \quad n \quad l \\ k \ n \quad n \quad u \quad n \quad k \ n \quad n \quad l \quad l \quad n \end{array} \\
\\
u \quad l \quad l \quad ul \quad u \quad n \quad un \ l \ n \quad lv \\
\\
\begin{array}{c} l \quad n \quad v \quad v \ n \quad v \quad l \quad n \\ un \ l \ n \quad l \ v \quad n \quad u \quad n \quad u \quad k \quad l \quad u \\ n \quad n \quad n \quad n \quad l \quad u \quad n \quad u \quad n \end{array} \\
\\
\begin{array}{c} n \quad n \quad l \quad l \quad k \ n \quad u \ v \ l \ n \quad n \quad n \\ n \quad n \quad n \quad l \quad n \quad l \quad n \quad v \quad n \end{array} \\
\\
\begin{array}{c} ul \quad l \quad n \quad l \quad u \quad n \quad ul \quad l \quad n \quad n \quad u \\ u \quad n \quad n \quad k \ n \quad l \quad n \quad n \end{array} \\
\\
\begin{array}{c} n \quad n \quad l \quad u \quad l \quad n \quad v \quad u \\ n \quad l \quad l \quad l \quad l \ n \quad n \quad n \quad v \quad u \\ n \quad l \quad l \quad l \ n \quad 3 \quad n \end{array} \\
\\
\frac{n \quad l \quad k \ n}{3 \quad m} \\
\\
m \quad m
\end{array}$$

3.3 Halting Problem for the Distributed Algorithm

n n n l n n l n un l n ZS n n l
l v u u nv n u n n v l
n n l l l v n l n
nv lv l l n un n n n n u l n
l ll v n u ll n u n n n n
n n l k n l n v n
n l n ll n n n u k n
n nv lv n l n u l k n n
n l u un n v
un k n n n u n nl n
n n l l k n u un n u l n
k n n n n
u n l l ZS n n un n u v u
k

Conclusions

v k n ln n
l n n u n n ZS n n v
n v n v n n n v
n n n u n n n PT n k n
l v l ZS n v n n ll k ll
n l n n v n n n nu
n n l n n n
v n v u n nv n n
u n u n n ZS n n k
ul n l n n n v l l l u n n
l l n l u l l n n PT n
nn n u n ul nn n n
u k k n n n n n
v n l n n u u n n
n n n l n l n n v l
ZS n

References

m m
m
mm m
Computer Architecture News (Theoret. Comput. Sci.

		<i>Proceedings STACS'93</i>	m	m	<i>Lect. Notes in Comput.</i>
<i>Sci.</i>	0	0			
m		mm			<i>Advances in Petri Nets'92</i>
	m	0			<i>Lect. Notes in Comput. Sci.</i>
		m			<i>Advances in Petri Nets'90</i>
	m				<i>Lect. Notes in Comput. Sci.</i>
		<i>Tile Logic for Synchronized Rewriting of Concurrent Systems</i>			
		m		m	
					m
	m				
	<i>Proceedings ASIAN'98, 4th Asian Computing Science Conference</i>				m
	<i>Lect. Notes in Comput. Sci.</i>				
					m
					m
	<i>Proceedings CTCS'99, 8th conference on Category Theory and Computer Science</i>		m		<i>Elect. Notes in Th. Comput. Sci.</i>
0					m
	m		m		<i>Proceedings EXPRESS'97, 4th workshop on Expressiveness in Concurrency</i>
	<i>put. Sci.</i>			m	<i>Elect. Notes in Th. Com-</i>
		<i>Proceedings WADT'97, 12th workshop on Recent Trends in Algebraic Development Techniques</i>		m	<i>Lect. Notes in Comput. Sci.</i>
	0				
			m		
	<i>Inform. and Comput.</i>				m
			m		000
					<i>Proceedings Int. Colloquium on Petri Net Technologies for Modelling Communication Based Systems</i>
	0				
				mm	
					m
	m				<i>Proceedings CONCUR'99, 10th International Conference on Concurrency Theory</i>
	<i>Comput. Sci.</i>			m	<i>Lect. Notes in</i>
		m		m	
			m		
m					
	<i>Proceedings LICS'95, 10th Annual IEEE Symposium on Logic In Computer Science</i>				
		0		m	
					<i>Inform. and</i>
	<i>Comput.</i>		m		
					m
					<i>Theoret. Comput. Sci.</i>
			(0
0					m
	<i>Theoret. Comput. Sci.</i>		0		

	<i>Kommunikation mit Automaten</i>	..	m
m			
0	m	..	m
	<i>Petri Nets: An Introduction</i>
m			
	m		m
	<i>J. Comput. and System Sci.</i>		
		m	<i>J. Comput. and System</i>
<i>Sci.</i>			
	m		m
	<i>Proceedings 12th International Workshop on Graph-Theoretic</i>		
<i>Concepts in Computer Science</i>	m	<i>Lect. Notes in Comput. Sci.</i>	
		<i>Proceedings Advanced Course</i>	
<i>on Petri Nets</i>	m	<i>Lect. Notes in Comput. Sci.</i>	

i	i	1	..	2	i i i	1
1	n		i n	ll	illi	n
	illi				ciardo, radu @cs.wm.edu	
	2		n l		n	
		n			luettgen@icase.edu	

Abstract.

y - - - i i i y i i y
-w y y y i i y i
i i y i i y y i y i
y i i y i y y i y
i i w i i y y i y iz
w i y yi i i i i y
y y w i i i i i i y
y i w i i i i y i i
y y i i i y i i
i y
* i i n l n i n ini i n
n n il in i n in i n in
n i li i n in n n in in
n l n n

i n n i l " n n i ini n
 - i iz
 0²⁰ i i y w
 i y 0 i i 3 i y
 i i i y i y y i -
 i i i y i y i i
 i y i i w i y
 i i y i i w i -
 i w i i i
 iz 0⁶⁰ 0⁶⁰⁰ i y
 w i
 i 0 wi 000 i
 wi i i i k k
 - i i
 i i i i i y y y i
 - i i i y y
 i i i i
 i y i y i y
 y - ki i y i i i
 w i i i y w i -
 i i i i i i y y
 i i y i i - y i -
 i i i i y i y -
 i i - ki w i-
 i w i i i
 y y ik i i i
 ik i y i y i i
 i i i i i -
 y i i i w i i
 i i i i w -k w i
 w i i i y i i y i
 y i i y i yi
 k i i i i y
 i i i i - k 0

2 Structured State Spaces and MDDs

i y i - y y y i w-
 y i i i i w i -
 w

$$\begin{aligned}
& \text{in} \quad \text{li} \quad \text{n} \quad \text{in} \quad \text{n} \quad \text{n} \\
& \text{i} \quad \text{i} \quad \text{wi} \quad \text{i} \quad \mathcal{P} \quad \text{i} \quad \mathcal{E} \\
& \text{i i i} \quad \text{ki} \quad s_0 \quad \mathbb{N} \quad \text{i} \quad \text{i} \quad \text{w} \\
& \quad e \quad \quad \quad s \quad \quad \quad s \\
& \quad \quad \quad s \quad \text{i} \quad \quad e \quad y \quad \mathcal{N}^{e,s} \quad \mathcal{N}^{e,s} \\
& \quad e \text{i} \quad \quad \text{i} \quad s \quad \text{wi} \quad \text{i} \quad \text{i} \quad \quad \text{i} \quad \mathcal{N}^{\text{i}} \quad \text{i} \quad y \\
& \text{i} \quad \quad \text{i} \quad \quad \text{i} \quad \quad \quad \mathcal{N}^{e,s} \quad \text{i} \\
& \quad \quad \quad \text{i} \quad \text{w} \quad \mathcal{N}^{e,s} \quad \text{i} \quad \text{i} \\
& \quad \text{i} \quad \text{i} \quad \text{i} \quad \text{i} \quad \quad \quad \text{i} \\
& \text{i} \quad \quad y \quad \quad \quad \text{i} \quad \text{i} \quad \text{i} \quad s_0 \quad \text{ii} \quad \text{i} \\
& \quad \quad \quad - \quad \quad \text{ii} \quad y \quad \text{i} \quad \text{i} \quad \text{i} \quad s \quad \mathcal{N}^{e,s} \\
& y \quad \quad e \quad \mathcal{E} \\
& \quad \text{i} \quad \quad \text{i} \quad \quad \quad \text{i} \quad \quad \text{i} \\
& \quad \text{ii} \quad \quad \text{i} \quad K \quad \quad y \quad \text{i} \quad \text{i} \quad \text{i} \quad \quad \mathcal{P}^{\text{i}} \quad K \\
& \text{i} \quad \text{i} \quad \text{i} \quad \quad \text{ii} \quad \quad \quad s \quad \quad \text{i} \quad K \quad \quad \text{i} \\
& s \quad \quad \quad s_K, s_{K-1}, \dots, s_1 \quad \quad \text{kw} \quad \quad \text{i} \quad \text{wi} \\
& \quad \quad \quad \text{i} \quad \text{w} \quad \quad \text{i} \quad \quad \text{i} \\
& \quad \text{ii} \quad \quad \mathcal{P} \quad \quad \text{i} \quad y \quad \quad \quad \text{w} \quad \text{i} \\
& \quad \quad \text{i} \quad \mathcal{N} \quad \quad \text{w} \quad \text{i} \quad \quad - \quad \quad K \quad \quad \text{i} \\
& \text{i} \quad \mathcal{N}^{e,s} \quad \mathcal{N}_K^{e,s_K} \quad \mathcal{N}_{K-1}^{e,s_{K-1}} \quad \mathcal{N}_1^{e,s_1} \quad \quad e \quad \mathcal{E} \\
& s \quad \quad \quad \text{i} \quad \quad \text{i} \\
& \text{i} \quad \quad \quad \quad \quad \quad \quad k \quad \quad s_{k,0}, s_{k,1}, \dots, s_{k,N_k-1} \quad \quad \text{i} \quad y \\
& \quad \quad y \quad \quad \text{ii} \quad \quad \text{i} \quad \text{w} \quad N_k \quad \mathbb{N}^{\text{i}} \\
& \quad \quad \text{i} \quad k \quad \quad \text{i} \quad \text{i} \quad \quad \text{i} \quad \text{ii} \quad \text{i} \quad \quad \text{ii} \\
& \quad \quad \text{i} \quad \quad \text{w} \quad \quad \quad \text{i} \quad \quad k \quad \text{i} \quad \text{i} \quad \text{i} \quad \quad y \\
& \quad \quad \quad k \quad \quad \text{i} \quad y \quad \quad y \text{i} \\
& \quad \quad \text{w} \quad \quad \quad \text{i} \quad \quad \quad \text{i} \quad \quad \text{ii} \\
& \text{i} \quad \quad k \quad \quad \text{i} \quad \text{w} \quad \text{i} \quad \text{iyi} \quad \text{wi} \quad \quad 0, \dots, N_k - \\
& \quad \quad \quad y \\
& \quad \quad f \quad 0, \dots, N_K - \quad \quad 0, \dots, N_1 - \quad - \quad 0, \quad \quad y \\
& f \quad s_K, s_{K-1}, \dots, s_1 \quad \quad \text{i} \quad \quad y \text{i} \quad s_K, s_{K-1}, \dots, s_1
\end{aligned}$$

Multi-valued Decision Diagrams.

$$\begin{aligned}
& \quad \text{i} \quad \text{i} \quad \quad \text{i} \\
& f \quad 0, \dots, N_K - \quad \quad 0, \dots, N_1 - \quad - \quad 0, \dots, M - \\
& \text{w} \quad K, M \quad \mathbb{N} \quad N_k \quad \mathbb{N} \quad K \quad k \quad \quad M \quad \quad N_k \\
& \quad K \quad k \quad \quad \text{i} \quad f \text{i} \quad \quad \text{i} \quad \quad \text{i} \quad \text{i} \quad \text{wi} \\
& \quad \text{k} \quad \text{w} \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{i} \\
& M \quad \quad \quad \text{i} \quad \text{i} \quad \quad \text{i} \quad \quad \quad \text{i} \quad \quad \text{i} \\
& \quad \text{ii} \quad y \text{i} \quad \quad \text{i} \quad \quad \quad y \\
& \quad \text{i} \quad \quad - \quad \quad \text{i} \quad \text{w} \quad \quad \text{ii} \quad \quad \text{ii} \quad \quad \text{i} \\
& \quad \quad \text{w} \quad \quad \text{i} \quad a \quad b \quad c \quad \text{i} \quad k \quad \quad 0, \quad , \\
& K \quad 3 \quad N_1 \quad N_2 \quad N_3 \quad M \quad 3 \quad \text{i} \quad \quad \text{wi} \quad \text{i} \quad \text{i} \quad y \\
& \quad \quad \text{i} \quad \quad y \quad \quad \text{i} \quad \quad \text{i} \quad \quad \text{w} \quad \text{i} \quad \text{i} \\
& \quad \quad \quad \text{wi} \quad \text{i} \quad \quad \quad \text{w} \quad \quad \text{i}
\end{aligned}$$

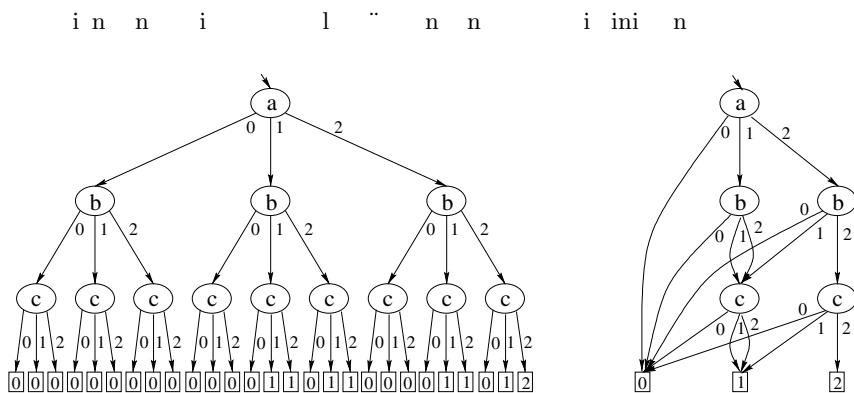
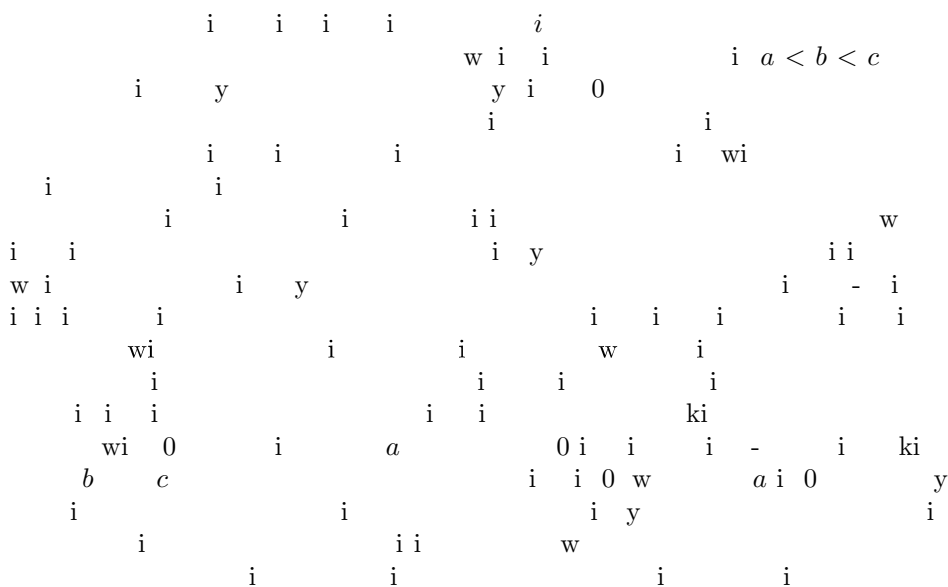


Fig. 1.



Data Structures for MDDs.

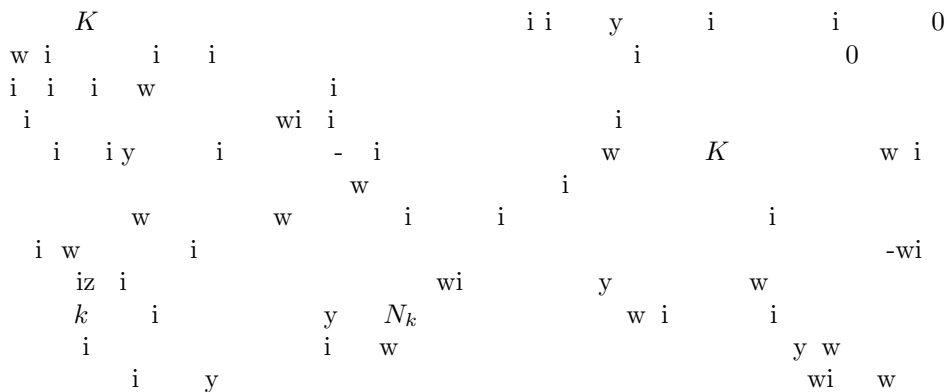


Table 1. i

<i>Union</i> in <i>p mddAddr</i> , in <i>q mddAddr mddAddr</i>	
1. if <i>p</i> , or <i>q</i> , return , ;	deal with the base cases first
2. if <i>p</i> , or <i>p q</i> return <i>q</i> ;	
3. if <i>q</i> , return <i>p</i> ;	
4. <i>k Max p.lvl, q.lvl</i> ;	maximum of the levels of <i>p</i> and <i>q</i>
5. if <i>LookUpInUC k, p, q, r</i> then return <i>r</i> ;	found in the union cache
6. <i>r CreateNode k</i> ;	otherwise, the union needs to be computed in <i>r</i>
7. for <i>i</i> to <i>N_k -</i> do	
8. if <i>k > p.lvl</i> then <i>u Union p, q dw i</i> ;	<i>p</i> is at a lower level than <i>q</i>
9. else if <i>k > q.lvl</i> then <i>u Union p dw i, q</i> ;	<i>q</i> is at a lower level than <i>p</i>
10. else <i>u Union p dw i, q dw i</i> ;	<i>p</i> and <i>q</i> are at the same level
11. <i>SetArc r, i, u</i> ;	make <i>u</i> the <i>i</i> -th child of <i>r</i>
12. <i>r CheckNode r</i> ;	store <i>r</i> in the unique table
13. <i>InsertInUC k, p, q, r</i> ;	record the result in the union cache
14. return <i>r</i> ;	

$$\begin{array}{c}
 y \\
 i \quad w \quad w \quad i \quad , \quad q \quad i \quad - \quad i \\
 i \quad i \quad q \quad i \quad i- \quad i \quad q \quad i \quad y \quad w \\
 0,0 \quad 0, \quad i \quad i \quad 0 \quad 0 \quad i \quad y
 \end{array}$$

The Union Operation on MDDs.

$$\begin{array}{c}
 i \quad i \quad y \quad i \quad i \quad w \quad i \quad k \quad w \\
 w \\
 y \quad i \quad i \quad i \quad w \quad i \quad i \quad y \quad i \quad i \quad i \quad i \\
 w \quad i \quad i \quad y \quad yz \quad w \quad i \\
 i \quad k \quad w \quad i \quad i \quad i \quad y \quad i \quad 0 \\
 i \quad k \quad i \quad i \quad i \quad i \quad i \quad i \quad 0 \\
 i \\
 i \quad i \quad i \quad i \quad i \quad i \quad 3 \\
 w \quad 0,0 \quad 0, \quad - \\
 i \quad y \quad k > 0 \quad - \quad i \quad i \quad kw \\
 i \quad p \quad q \quad i \quad y \\
 i \quad i \quad i \quad wi \quad w \quad ki \\
 w \quad i- \quad i \quad i \quad i \quad y \quad i \quad y \quad i \quad i \quad i \quad i- \quad i \\
 p \quad i- \quad i \quad q \quad 0 \quad i < N_k \quad i \quad w \\
 k \quad i \quad i \quad i \quad y \\
 y \quad i \quad i \quad i \quad i \quad y \quad i \\
 i \quad r \quad i \quad k \quad y \quad i \quad i \quad r \quad r \quad i \quad - \\
 y \quad r \quad r \quad i \quad i \quad r \quad i \quad i \\
 r \quad wi \quad i \quad i \quad r \quad i \quad i \\
 i \quad i \quad i \quad i \quad y \quad i \quad 3 \\
 i \quad y \quad i \quad i \quad i \quad y \quad i \quad y \quad i \quad 3
 \end{array}$$

Table 2. i - i

<i>MDDgeneration</i> in $m : \text{array } \dots, K \text{ of int } mddAddr$	
1. for k to K do <i>ClearUT</i> k ;	clear unique table
2. $q \leftarrow \text{SetInitial } m$;	build the MDD representing the initial state
3. repeat	start state-space exploration
4. for k to K do <i>ClearUC</i> k ;	clear union cache
5. for k to K do <i>ClearFC</i> k ;	clear firing cache
6. <i>mddChanged</i> \leftarrow false;	<i>true</i> if MDD changes in this iteration
7. foreach event e do <i>Fire</i> $e, q, mddChanged$	fire e , add newly reached states
8. until <i>mddChanged</i> \leftarrow false;	keep iterating until fixed point is reached
9. return q ;	return MDD representing the reachable state space

MDD-based State-space Construction.

MDD-based State-space Construction.

3 The Concept of Event Locality

$$\begin{array}{ccccccc}
i & & - & i & & y \\
& i & w & i & y & y & y \\
& i & y & i & & i & y \\
e & i & i & & k- & & i- \\
i & i & k & i & s_k & s_k & s_{K}, s_{K-1}, \dots, s_1 \\
s & s_K, s_{K-1}, \dots, s_1 & \mathcal{N} & e, s & wi & e & k- \\
& k & & & y & i & k & i & i \\
& & k & wi & i & i & & & \\
e & & e & & i & i & & wi & e- \\
& e & i & i & & y & k & i & yi & K & k > & e \\
e > k & & wi & e & i & & i & & i & y & w
\end{array}$$

$$\begin{aligned}
& \text{in} \quad \text{li} \quad \text{n} \quad \text{in} \quad \text{n} \quad \text{n} \\
& e \quad i \quad e \quad y \quad y \quad i \quad i \quad - \\
& \quad \quad \quad y \quad i \quad i \quad y \quad K \quad e \quad l_k \quad i \quad y i \\
& \quad \quad \quad k \quad i \quad i \quad l_k \quad i \quad y i \\
\mathcal{N}_k \quad l_k, s \quad \text{df} \quad \bigcup_e : First(e)=Last(e)=k \quad \mathcal{N}_k \quad e, s \quad s \quad i \quad i \quad - \\
& \quad \quad \quad y \quad i \quad i \quad y \quad i \quad i \quad y \quad - \\
& \quad \quad \quad i \quad i \quad i \quad i \quad i \quad i \quad i \quad - \\
& \quad \quad \quad i \quad y \quad i \quad e \quad w \\
& e \quad e \quad i \quad e \quad y \quad w \\
& \quad \quad \quad i \quad y \quad i \quad e \quad y \quad w \\
& e \quad i \quad y \quad w \quad e \quad y \\
& i \quad i \quad i \quad w \quad i \quad i \quad i \quad - \\
& i \quad w \quad y \quad i \quad i \quad i \quad - \\
& \quad \quad \quad y \quad e \quad w \quad y \quad i \quad i \quad - \\
& i \quad i \quad i \quad i \quad i \quad i \quad i \quad - \\
& i \quad w \quad y \quad w \quad i \quad i \quad i \quad K- \quad i \quad i \quad y \\
& i \quad w \quad w \quad w \quad i \\
& \quad \quad \quad i \quad y \quad e
\end{aligned}$$

Implicit Roots.

$$\begin{aligned}
& w \quad e \\
& i \quad y \quad w \\
& i \quad i \quad y \quad i \quad i \\
& \quad \quad \quad w \quad i \quad p \quad i \quad e \quad i \quad i \\
& q \quad k \quad i \quad y i \quad e > k \quad e \quad i \quad i \quad i \quad i \quad i \\
& i \quad i \quad - \quad i \quad y \quad w \quad -i \quad i \quad i \quad i \\
& \quad \quad \quad e \quad w \quad w \quad i \quad y \quad i \\
& e \quad w \quad i \quad i \\
& i \quad i \quad i \quad q \quad i \quad q \quad i \\
& p \quad i \quad q \quad i \quad i \quad i \quad y \\
& i \quad i \quad i \quad i \quad e \\
& e \quad y \quad k i \quad w \\
& \quad \quad \quad e \quad i \quad i \quad i \\
& i \quad i \quad e \quad w \quad y \quad i \quad i \\
& i \quad p \quad p \quad - \quad w \quad i \quad i \\
& \quad \quad \quad w i \quad i \quad e \quad i \quad y \quad i \\
& \quad \quad \quad k \quad i \quad w \\
& i \quad i \quad i \quad y \quad w \quad w \quad i \quad w \quad i \\
& \quad \quad \quad e \quad y \quad i \quad e \quad i \quad i \quad i \quad i \\
& y i \quad w \quad i \quad y \quad i \quad i \\
& i \quad w \quad i \quad i \\
& \quad \quad \quad w \\
& i \quad i \quad i \quad i \quad w \quad i \\
& \quad \quad \quad y \quad y \quad w \quad i
\end{aligned}$$

The diagram illustrates the insertion of explicit nodes to deal with implicit roots at level $First(e)$. It consists of two parts: a left diagram showing a tree structure and a right diagram showing a detailed view of the insertion process.

Left Diagram: A tree structure with a root node labeled '1' in a box. A shaded region represents a subtree. Two horizontal dashed lines are labeled $First(e)$ and $Last(e)$. Nodes p and p' are located above the $First(e)$ line, and node q is located below the $Last(e)$ line. Arrows indicate a path from p to p' to q .

Right Diagram: A detailed view of the insertion process. It shows a shaded region with a wavy bottom edge. An oval labeled 'implicit root' is positioned above the $First(e)$ line. Nodes p and p' are located above the $First(e)$ line, and node q is located below the $Last(e)$ line. Arrows indicate the insertion of explicit nodes p and p' to deal with the implicit root at level $First(e)$.

Explicit nodes need to be inserted in order to deal with implicit roots at level $First(e)$.

i w i
i i
i w y y
k w

In-place Updates.

in li n in n n
 e i
 i i
 i - i y iz
 i i i y i -
 i i i i i
 i y i e i i *K*- i i i i
 i - i y w
 e i i
 izi i i i y w i y
 i i i i y - - i i y
 ii i y w i i y
 i y y i i i i
 i i i i i y i i
 i i i i i i
 w i iz i i i
 y izi

4 Improving Cache Management and Iteration Control

i y w i i i -
 i

Intelligent Cache Management.

i i - w
 i y i w k i i
 i i i i -
 i i y i - w ii
 i i w
 y i e i e w y i i i
 i i i i i i i
 i i i w i y fl
 i fl w y i i fl i -
 i w w i wi i yi i i
 i i y k i i
 i i w ii y i i
 0 i
 i y 3 w yi i i
 ii y w i i iz i i i
 w i i i i i *p, q*
r *p, r* *q, r* wi *r* w
 ii iz i i y i

i n n i l " n n i ini n
 y p,q i i i i i i
 i y i i i i i
 wi w i i e i p k
 j $\mathcal{N}_k e, i$ y
 p i y
 p j r p i, p j w i
 w p j i i i p
 w e i p i y
 e i i i w
 p j p i, p j i w i p
 p i, p j i i i p i, r r y i
 r i i i i w i i i i

Advanced Iteration Control.

i y w
 i i i i i -
 i i i i 0 i i y wi i
 i i i i i i i
 i w w i i
 i i i i y i e i yi e
 e i l_1 i k i
 y i y i l_k e
 i yi e k e i k w
 y izi y k i
 i i i w y i i
 i i i i y i i
 y w ki i y i i i w
 i i y i w i i
 i i - i i w i i
 y i i y i w i
 i yi i w i y izi i
 i i w i w i i y
 i y w i y i i
 yi i i y i -
 s w i i i i i y izi
 i s i i i
 i i w i i - i -
 i i i i i iz

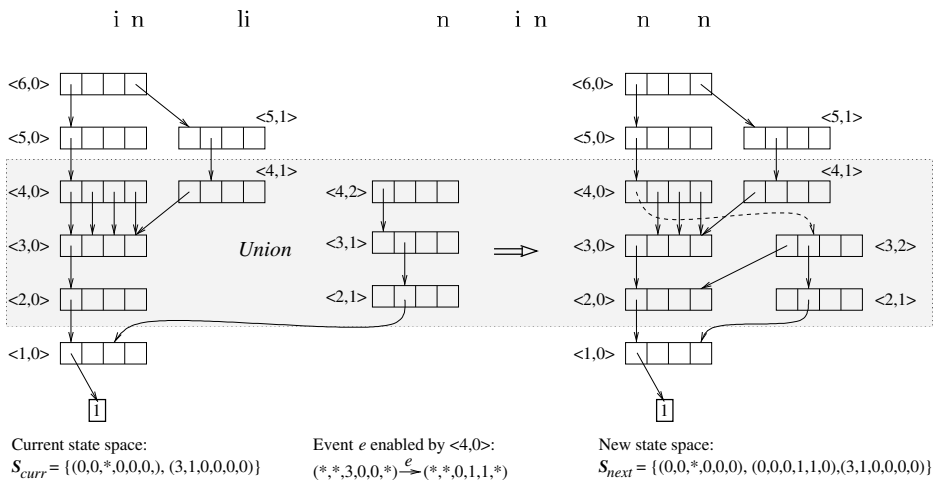
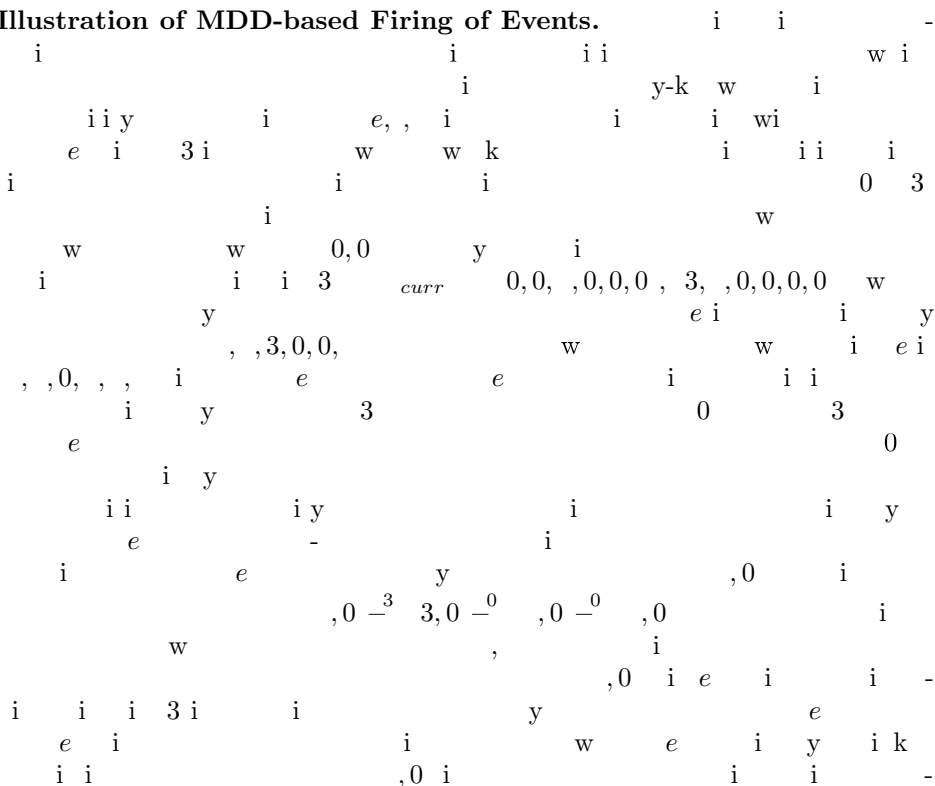


Fig. 3.

5 Details of the New Algorithm

Illustration of MDD-based Firing of Events.



i n n i l " n n i ini n
 i , i y i w i i
 i i $\alpha, 0, , , \beta$ w
 α i y i , 0 w β i
 0- , 0 α β y -
 0, 0 0 i y w y
 i e i , 0 i add 0, 0, 0 , , 0 i y 0- w
 , 0 i i i
 3, 3, 0 3, w i i i w
 3, i i i i 3 i
 next i 0, 0, , 0, 0, 0 , 0, 0, 0 , , 0 , 3, , 0, 0, 0, 0 -
 i e i i i i i i i i y
 e i i i i y
 i i i iz i

Further Implementation Details.

i i fl i i y
 i i i i -
 w i -
 i i i
 y i i y y i i i y
 i i y i y i w i i y-i k
 i wi i i k i y
 i i i iz
 y k y i i i i dw- y i
 w i i i
 i k y wi fl i w
 i i i wi i i y
 i y k y i i i i y
 i i y i i i i i w
 i i i i wi
 - i i i w i wi
 wi k y i w i i i
 i i i i
 i - i i i
 w i w i i y i
 i k w i y y i
 i i y i w
 fl y i
 q y p w

Correctness of the Algorithm.

i i w i w i
 i wi y

The Petri net for the Dining Philosophers problem consists of the following places and transitions:

- Places:**
 - $GoEat_i$ (top center, rectangle)
 - $WaitLeft_i$ (top left, circle)
 - $WaitRight_i$ (top right, circle)
 - $GetLeft_i$ (middle left, rectangle)
 - $GetRight_i$ (middle right, rectangle)
 - $Idle_i$ (center, circle with a token)
 - $HasLeft_i$ (bottom left, circle)
 - $HasRight_i$ (bottom right, circle)
 - $Fork_{(i+1) \bmod N}$ (bottom left, dashed circle with a token)
 - $Fork_i$ (bottom right, dashed circle with a token)
- Transitions:**
 - $GoEat_i$ (top center, rectangle)
 - $GetLeft_i$ (middle left, rectangle)
 - $GetRight_i$ (middle right, rectangle)
 - $Release_i$ (bottom center, rectangle)
 - $Fork_i$ (bottom right, dashed circle)
- Flow:**
 - $GoEat_i$ leads to $WaitLeft_i$ and $WaitRight_i$.
 - $WaitLeft_i$ leads to $GetLeft_i$.
 - $WaitRight_i$ leads to $GetRight_i$.
 - $GetLeft_i$ leads to $HasLeft_i$.
 - $GetRight_i$ leads to $HasRight_i$.
 - $HasLeft_i$ and $HasRight_i$ lead to $Release_i$.
 - $Release_i$ leads to $Idle_i$.
 - $Idle_i$ leads to $GoEat_i$.
 - $Fork_i$ leads to $HasRight_i$.
 - $Fork_{(i+1) \bmod N}$ leads to $HasLeft_i$.

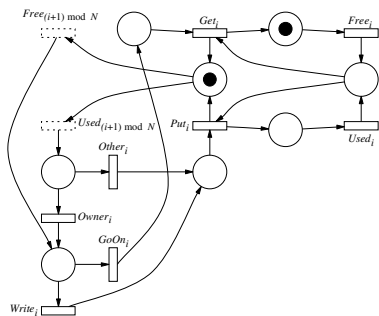
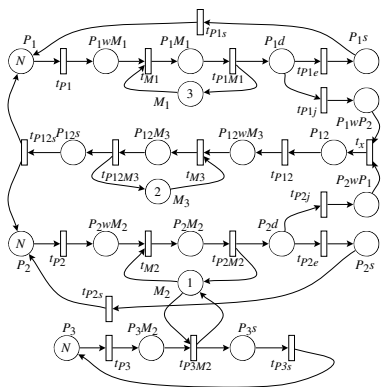
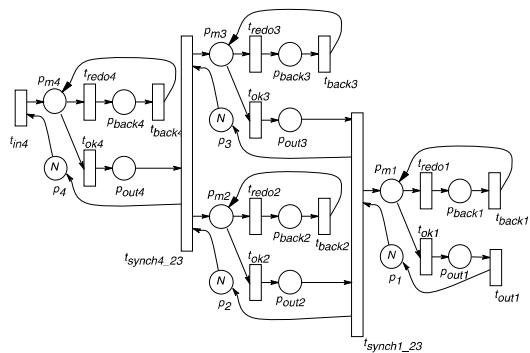


Fig. 4. $\begin{array}{ccccccc} & i & & i & & i & i & i \\ y & & i & & w & & & i & w & i \\ & & & & & & & & & \\ & y & & i & & i & & i & i & - \\ i & & i & & & & y & & & i \\ & & i & i & i & & & & & \\ fl & & & & & & & & & \\ w & & i & i & y & k & 0 & & N & i & i \\ i & & & & i & & k & & i & & w \\ i & i & y & k & & & & & & i & i \\ & iz & y & & N & i & i & i & k & i & i \\ & y & & i & i & i & & & & w & \\ i & i & & i & w & i & w & i & & & \\ & & -i & & & w & i & w & i & & \\ y & i & & & & i & & & & & P_1 M_1, M_1 \\ P_{12} M_3, M_3 & & P_2 M_2, M_2 & & & & & & & & p_{mX}, p_{backX}, p_{outX}, p_X \\ X & , & 3, & & i & y & & & & & \\ & 3 & & & & & & & & & \\ w & & -k & w & i & i & & i & & w & i \\ & 00 & z & & i & w & k & i & wi & & y \end{array}$

i n l i n i n n

i N w i iz

w i i i

i w i k i

i i i y i i y i

i i - w i - i i k -

y i - k

w i - y -

w i y i y i i

i 3 33 - i wi i i

y i i i

i y y i w i i

i y i w i i

w i w i i

w i w i i

w i f l i i y i

i y i i z i i i i 3

i i i i i y i

w - i w i i

i i i i y i N

- i i i i w i -

i i y w i w w i i

w i i i i y w k w i

y i i i y i w

w w N i y

i i i - i wi i i y

i i y i i i

w i w i - i i

i i i i k-k i

y i y

N i i i

i i i i i

i i y wi

y y i y y-

Table 3. i i i i

			in				n				
N	S	n l	n		i	i	n		i	i	
		n									
	. 6										
	. 31										
	. 62										
	. 125										
	. 188										
	. 250										
	. 313										
	. 376										
	. 438										
	. 501										
	. 564										
	. 626										
	. 9										
	. 20										
	. 31										
	. 41										
	. 52										
	. 6										
	. 9										
	. 11										
	. 12										
	. 13										
	. 17										
	. 19										
	. 21										
	. 6										
	. 9										
	. 10										
	. 11										
	. 12										
	. 13										
	. 14										
	. 16										

i i y i w i - i i i
w i y y i i i
i wi i i i i
i i w y i y y i i i i
i i i k i i i
w i y w w i k
i w i i i y i i

i n n i l " n n i ini n
 y i i i i i i
 i i i wi iii
 i i i y i - y i
 w y y w y y -
 0 i 0 - -
 i i i i y y i
 y i wi i i i i w i y
 i i 3 i i w k
 i i i y i i i -
 i w yi i i i i
 w k i i i i y
 y w i i i i i
 i i i i w i -
 w i i y i i y i i w
 y - i i i
 i w i i i i

8 Conclusions and Future Work

i i i
 y y i i w k y -
 i y i i- i i i
 i w k i y i i y i y
 y i i i i -
 i i i i
 w -k w i i i y w i i i i -
 i y i i wi y
 i i y k w i i
 y i i ki i y
 i i w k w w ik i
 i i i i i w i fl w
 i i i i i w y -
 i i i i i k i
 y
 i y k y i y w i iz i
 - y i i - y i

Acknowledgments. w ik k i y
 i i

References

- n l i l n n in ni l i n *IEEE Trans. on Computers*
- n li l n ni l i n i in i i n i *ACM Computing Surveys*
- l i i l in n *PNPM '97*
- i
- l in ²⁰ n ill n ill n n li
- i " n n i ini n i n n i n
- n n n n in in n i li in
- i n in i l i n n i n n l
- li ili n i in n *IPDS '96* i
- i n in l n i l
- n *Tools '97* l *LNCS* in l
- i n il n n n l i n
- n li i i n n n i
- li i n in i n n in in
- l il n n l i in in l in n
- CAV '93* l *LNCS* in l
- l n l *Model Checking*
- n " n n n in i in
- i i n n ni
- i *Partial-order Methods for the Verification of Concurrent Systems – An Approach to the State-explosion Problem* l *LNCS* in l
- n n " n i i n l ini i i n ni
- in in i i n *FAC*
- l nn l in *IEEE Trans. on Software Engineering*
- ill n n ni nni in n lli l i l
- i i n i n li i n *Multiple-Valued Logic*
- i l n l i *IEEE Trans. on Software Engineering*
- in n i i n ili n i n n in
- i i n i n *ICATPN '99* l *LNCS* in l
- in i n l i n li i n *Proc. of the IEEE*
- n ll i n n in li n l i
- i n *DATE '98* i
- i ll n i i n n l i in
- l n ni l i n n *ICATPN '94* l *LNCS* in
- l
- i n n i i n i ili n l i n *ICCAD '95*
- i

i n n i	l " n n	i ini n
l i	n n	l i n l n <i>CAV '90</i>
ni i	l i n n n	l n
n l n	l in i ni	n l
n	n i l in	n <i>ICATPN '96</i>
l <i>LNCs</i>	in l	
n n	ll n ll l	n i n n
<i>PPOPP '97</i> l	<i>ACM SIGPLAN Notices</i>	

Designing a LTL Model-Checker Based on Unfolding Graphs

- i ou u ¹ i i ¹ i oi u ²

¹ ni i n n

couvreur, grivet@labri.u-bordeaux.fr

² ni i i i i i n

poitrenaud@lip6.fr

Abstract. n i n n ni igning

n fl ing in gi ing

n ing g n g i n in

i n gni i n ing i in

n n n n fl n i n n n ing g

i i n i n in i

ign i n g i i n ing

n i i i n *dead* i

in i ni n

i i g in n ing

Topics. i ing in gi n ing

1 Introduction

u o i ki i o i i i i-

io i o i o o io u i

i o o i i o u i io u o i

i i o i o o ki o i k o

- o 22 i o ki o o i

i i i o o i u

i o oi o io o o o - -fl

i io 2 i o u io 0 2 2

o i i o u o i i io o i

- i u i i io 2

o o o i i i i i io i o -

io o i o o i o o i -

i o i o o i i i o u io

o i i o i i ki

o o i o u o io o o

o k ii o o i i

i o o i o o i o u o i o u -

io i io o o i o i io o

i o o i i o o i 2 2

n i i n i n n i n

i i u i i o o o - -fl o
ki o o i o o i u i u o i o
k o i i 2 o u io o u o o
o i o u ki o i ou o k o i u
i i i o o i 20 LTL i ou - i o o LTL
X u o o o u o u o o i io o
u u o o i o o i o i i i o u o o
o i i o i i i o u u o o
i u o o u io o 2 o i o u
u o i o i o i o o o u o
u o i i i io o u o o o i io o
o i o o i o o u i o
o - -fl o u io o u o i u o i
i o u u o o ki i u o i
o u i i o i o o i o o u i
u o o o i io i o i i o u-
o o o o i o i i io o
i u o i o i i o i i u o i o i o
o i i o i io i io i o i u u
o o i o o i o io o i i o i o
o u o i o o o i u k i
i i o i o u i i u io u o
u o i io
ou u o u o o i o o i o *dead dead*
o o i ki i o o i o i o o o
i io o u o u o
u io i u u o o i i o o i
i u o o o u io u o o o i io
o io o i *dead* o o i o u o u u
o u io o o k i o ou i o o i o i u i o -
i io o o i i i
ki o i i o u u o o
- v v i io i i i u o o i io o
o i o o i o
- v v i io i i u o o i io
o o i o o i o
- v i io i o i
i io o u o i io o o i o o i o
- v i io i o i
i io o o o u o i io o o i o o i o
u o i u i i io o o i io o
o ki o io o o u o i o -
u io o o i i i o i io o o u
u o o

o o i i i i i i
 o o i u o i io i io o u -
 i o i u o o o *LTl* *X* o u io i i io
 o i *LTl* *X* o u i u o i io o i u
 o k i o ou *dead* o i o o i io io i i o
 i io o o u i k o

2 Preliminaries

2.1 P/T System and Linear Temporal Logic

A B i o i i $PowerSet$ A o o u
 o A A B o o i o A o B i A
 ki i A B i i
 S *true, false* o o i io o S i AP o
 o o i io o S o i o o i io o o i io ui
 u ui oo o o $Prop$ AP i o o o i io
 u io Φ_{AP} i i S AP *true, false*
 m S , p AP Φ_{AP} m p p m
 i i $\langle N, m_0 \rangle$ N $\langle P, T, Pre, Post \rangle$ i P
 T i oi o i io Pre $Post$ i i
 i i P T m_0 i i i i ki i P io
 o i i u i i io t i
 o ki m i P m_-^t i p P m p Pre p, t
 i i o ki m m_-^t m i m_-^t p P m p
 $Post$ p, t $-Pre$ p, t m p o ki i o
 i i i ki m_0 i i io o i
 i i i o $Reach$ N, m_0 i ou
 i i i i
 u o i i i u o ki ρ m_0 m_1 $m_2 \dots$
 u m_0 i i i i ki o i 0, t T $m_i_-^t$ m_{i+1}
 i ou o o i o o i ki oo i-
 io i i u i o i i u u
 i i i i
 o i o o i io a o i o o i io o o
 ki P AP o o i o o i io o u
 ρ o Φ_{AP} ρ i i i o o AP *true, false*
 σ Φ_{AP} ρ Φ_{AP} m_0 Φ_{AP} m_1 Φ_{AP} $m_2 \dots$
 u i o o i o ou i io u
 o i o o i S u
 i o o f i o u o S u f o
 o i i o
 o i o o i io AP i o u
 2 f g o u o f f g Xf fUg

$$\begin{array}{c}
n \quad i \qquad \qquad \qquad i \quad n \quad i \quad n \quad n i \quad i \quad n \\
\\
i \qquad \qquad i o \quad o \qquad \qquad o \quad u \quad i \quad i \quad i \quad o \quad \sigma \quad x_0.x_1.x_2 \dots o \\
AP \quad \quad \quad true, false \quad \quad \quad u \qquad \qquad \qquad o \quad i o \quad \sigma \quad f \quad o \quad i \quad i \\
u \quad o \qquad \qquad o \quad u \quad f \quad o \quad i \quad i \quad o \quad \sigma \quad \quad i \quad \sigma_i \quad o \quad \quad u \\
o \quad \sigma \quad i \quad \quad x_i \qquad \qquad i o \quad i \qquad \qquad i \quad u \quad i \qquad \qquad o \quad o
\end{array}$$

$$\begin{array}{c}
\sigma \quad \quad i \quad x_0 \quad \quad o \quad \quad AP \\
2 \quad \sigma \quad \quad f \quad i \quad \sigma \quad \quad f \\
\sigma \quad \quad f \quad g \quad i \quad \sigma \quad \quad f \quad \quad \sigma \quad \quad g \\
\sigma \quad \quad X f \quad i \quad \sigma_1 \quad \quad f \\
\sigma \quad \quad f U g \quad i \quad i \quad 0 \quad \sigma_i \quad \quad g \quad \quad j < i \quad \sigma_j \quad \quad f
\end{array}$$

$$\begin{array}{c}
\qquad \qquad \qquad o o \quad \quad o \quad \quad o \qquad \qquad \qquad o \quad \quad true \quad \quad false \quad \quad o \\
u \quad \quad o \quad o \quad u \quad \quad \quad o \quad u \quad \quad o \quad \quad o \quad u \quad f \quad \quad o \quad \quad Sub \quad f \quad i \\
o \quad u \quad - \quad o \quad u \quad \quad \quad i \quad o \quad i o \quad i \qquad \qquad \quad o \quad o \quad u
\end{array}$$

2.2 Unfolding

$$\begin{array}{c}
i \quad u \quad \quad i o \quad u \quad o \quad i \qquad \qquad u \quad \quad i \quad \quad o \quad i o \qquad \qquad o \quad i o \qquad \quad - \\
\\
o \quad \quad o \quad x \quad o \qquad \qquad \quad x \quad \quad x \quad \quad o \qquad \qquad \quad i \qquad \qquad \quad o \quad \quad - \\
o \quad \quad \quad u \quad \quad o \quad o \quad x \quad i \qquad \qquad \qquad \qquad \quad i \quad o \quad i o \quad i \\
o \quad o \quad \quad \quad u \quad u \qquad \qquad \qquad \qquad \quad i o \quad o \quad \quad i \quad \quad o \\
o \quad \quad \quad i \qquad \quad N \qquad \qquad \quad i \qquad \qquad \quad x \quad \quad y \quad \quad o \quad o \quad \quad o \quad N
\end{array}$$

$$\begin{array}{c}
- \quad x \quad i \quad \quad u \qquad \qquad \quad o \quad y \quad \quad o \quad \quad x \quad \quad y \quad i \qquad \qquad i \qquad \qquad \quad o \quad x \\
\quad \quad o \quad y \quad i \\
- \quad x \quad \quad y \quad \quad i \quad o \quad f l i \qquad \quad o \quad \quad x \# y \quad i \qquad \quad i \quad \quad o \quad \quad i \quad i o \quad \quad i \\
\quad \quad \quad u \qquad \qquad \quad i \quad \quad o \quad x \quad \quad y \qquad \qquad \quad i \quad u \\
- \quad x \quad \quad y \quad \quad o \quad u \qquad \quad o \quad \quad x \quad \quad y \quad i \quad \quad i \quad o \quad u \\
\qquad \qquad \qquad \quad o \quad i \quad o \quad f l i
\end{array}$$

$$\begin{array}{c}
o \quad i \qquad \qquad \quad i \quad u \qquad \qquad \qquad \quad i \quad o \quad u \quad \quad o \quad i \quad o \\
o \quad \quad o \quad \quad i \quad \quad i \quad i o
\end{array}$$

Definition 1 (Occurrence and causal net). $N = \langle B, E, Pre, Post \rangle$

- N $o \quad u$
- $Pre, Post \quad B \quad E \quad 0,$
 - N v
 - v u
 - $f l \quad x \quad B \quad E, \quad x \# x$
 - $v \quad N \quad u$
 - v $u \quad u$

igning

n n ing

o u u i io i
v
i i u io o i o
o o u o u N o
Min N o i i o i i i o
o u u C o o u i i i
- o i x E, y C x y x C i o fli -
x, y C, x[#]y o o u N o Conf N i o
o u io MaxConf N o i i u io
o u io i i o u io i u io i o i
e i u o e i o e io
o o i io b o b o o u io b i b i u
o i o o o A B E A o
o i u io o o o u io o i
A o A i o i o u io
o o u o o i io o u i i i o fli - u
o u io o o u io C i o i u
o Cut C Cut C Min N C C
i i i o u i u io
h i o A o B h o PowerSet A
o B i o o X A, b B, h X b x X h x b
io o h k u i A ou o i i
i b i o o i u i i i o

Definition 2 (Homomorphism). $N \langle P, T, Pre, Post \rangle \xrightarrow{h} N \langle P, T, Pre, Post \rangle$
 $o \ i \ N \ N$

- $p \ P, h \ p \ P$
- $t \ T, h \ t \ T$
- $t \ T, p \ P, h \ t \ p \ Pre \ p, h \ t \ h \ t \ p \ Post \ p, h \ t$

o i o i io o o o o i
u o i

Definition 3 (Branching Process and Process). $N \langle P, T, Pre, Post \rangle \xrightarrow{m} N \langle B, E, Pr, Po \rangle \xrightarrow{h} U \ N \langle U, h \rangle \xrightarrow{i \ o} \langle N, m \rangle$

- $m \ h \ Min \ U$
- $e, e \ E, e \ e \ e \ e \ h \ e \ h \ e \ e \ e$

v $\langle U, h \rangle$ u v u
 $\langle U, h \rangle$ o $\langle N, m \rangle$

o u io o i o o o o o
i o $\langle U, h \rangle$ o $\langle N, m \rangle$ o o i o u -
io C u o u o o o i C i u ou i
o i io i i o i io o U i o o $\langle N, m \rangle$

u o i i i ki u o u
i o i u i ii o i o
o u i io

3 Automata Stuttering Property for $LT\bar{L} \setminus X$ Formula

$LT\bar{L} \setminus X$ i io o i u i i i o o i 20
o o i io o u u o o i o o i io i i
i io io o o i u o o u io o 2
o o u o u u o i io o i
o o o u o u o i i i io o u o -
o o o i io o o i o o i io o o u i
o i u o i i *dead* o i o o i io
o o i io ü i u o o $\langle Q, Acc, , q_0 \rangle$ o o i o -
o

- Q i i o
- Acc i i o i o i io
- Q $Prop$ AP $PowerSet$ Acc Q i i io io
- q_0 i i i q_0 Q

i i o σ $x_0.x_1.x_2 \dots$ o AP $true, false$ i
i io ü i u o o i i i i

ρ q_0 (X_0, A_0) q_1 (X_1, A_1) q_2 (X_2, A_2) \dots

u i 0 q_i, X_i, A_i, q_{i+1} X_i x_i a Acc, j i a A_j
u o o o u io i o u o u o o
o o u i io o i
i o o o i o o i i ui o
u i i o o u o o o o
o i u u o i u i u o o o i i 2
o i o o i io i i o u o o ou u
i o u o i u o

Theorem 1. f u \ddot{u} u
 Bu f v
 AP $true, false$ f v Bu f u

- Q $PowerSet$ Sub f
- f
- G Q \ddot{u} u Bu f G
 G
- $\langle G, X, A, H \rangle$ $Domain$ H $Domain$ G

$$\mathbf{n} \quad \mathbf{i} \qquad \mathbf{i} \mathbf{n} \quad \mathbf{i} \quad \mathbf{n} \quad \mathbf{n} \mathbf{i} \quad \mathbf{i} \quad \mathbf{n}$$

$$\begin{array}{c} v \quad G \\ \text{Domain } G \quad AP \quad \text{Sub } G \end{array} \qquad G \qquad \bigwedge_g G \, g$$

$$\begin{array}{c} \text{o} \quad \text{u} \quad \text{i} \quad LTL \quad X \quad \text{u} \qquad \text{u} \quad \text{i} \quad \text{o} \quad 20 \quad \text{i} \quad \text{i} \quad \text{o} \\ \sigma \quad x_0 \, x_1 \, x_2 \dots x_i \, x_{i+1} \dots \quad \sigma \quad x_0 \, x_1 \, x_2 \dots x_i \, x_i \, x_{i+1} \dots \quad \text{i} \\ \text{o} \quad LTL \quad X \quad \text{o} \quad \text{u} \qquad \text{o} \quad 2 \quad \text{o} \quad \text{i} \quad \text{i} \quad \text{o} \end{array}$$

$$\textbf{Theorem 2.} \quad \begin{array}{c} f \quad LTL \quad X \quad u \quad \sigma \quad x_0 \, x_1 \, x_2 \dots \\ v \quad AP \quad true, false \quad \sigma \quad x_{i_0} \, x_{i_1} \, x_{i_2} \dots \\ \sigma \end{array}$$

$$\begin{array}{l} - \quad k \quad i_k < i_{k+1} \\ - \quad k, \quad i \quad i_k \quad i < i_{k+1} \quad x_i \, \text{Domain}(f) \quad x_{i_k} \, \text{Domain}(f) \\ - \quad k \quad x_{i_k} \, \text{Domain}(f) \quad x_{i_{k+1}} \, \text{Domain}(f) \quad i \quad i_k \quad x_i \, \text{Domain}(f) \quad x_{i_k} \, \text{Domain}(f) \\ \sigma \, u \quad f \quad \sigma \, u \quad f \end{array}$$

$$\begin{array}{c} \text{o} \quad 2 \quad \text{u} \qquad \text{o} \quad \text{i} \quad LTL \quad X \quad \text{o} \quad \text{u} \quad \text{o} \quad \text{i} \quad \text{i} \quad \text{o} \\ \text{o} \quad \text{i} \qquad \text{o} \quad \text{i} \qquad \text{u} \quad \text{u} \quad \text{o} \quad \text{o} \quad \text{i} \\ \text{o} \, \text{o} \, \text{i} \, \text{i} \, \text{o} \qquad \text{i} \quad \text{u} \quad \text{i} \quad \text{i} \quad \text{o} \qquad \text{i} \\ \text{i} \, \text{i} \, \text{o} \quad \ddot{\text{u}} \, \text{i} \, \text{u} \, \text{o} \quad \text{o} \quad \text{o} \, \text{i} \, \text{i} \quad \text{i} \quad \text{o} \qquad \text{o} \, \text{o} \, \text{i} \quad \text{o} \\ \text{i} \qquad \text{o} \, \text{i} \, \text{i} \, \text{i} \qquad \text{i} \quad \text{o} \, \text{i} \, \text{i} \, \text{o} \quad \text{u} \\ \text{i} \quad \text{i} \quad \text{o} \quad \text{i} \quad \text{u} \, \text{o} \quad \text{o} \qquad \text{o} \, \text{i} \quad \text{i} \quad \text{i} \quad \text{x} \\ \text{i} \quad \text{i} \qquad F \quad \text{o} \quad \text{u} \end{array}$$

$$\begin{array}{c} x^\omega \quad F \\ \text{o} \qquad \text{i} \quad \text{o} \quad LTL \quad X \quad \text{o} \quad \text{u} \qquad \text{i} \, \text{i} \quad \text{u} \, \text{o} \quad \text{o} \\ \text{i} \quad \text{o} \quad \text{o} \quad \text{o} \qquad \text{u} \quad \text{i} \quad \text{o} \qquad \text{i} \quad \text{i} \quad \text{o} \\ \text{u} \, \text{o} \quad \text{o} \qquad \text{i} \, \text{i} \qquad \text{u} \quad \text{i} \quad \text{o} \, \text{i} \, \text{i} \, \text{o} \quad \text{o} \quad \text{o} \, \text{i} \\ \text{o} \, \text{o} \, \text{i} \, \text{i} \, \text{o} \quad \text{i} \qquad \text{i} \quad \text{o} \, \text{u} \quad \text{o} \quad \text{u} \, \text{o} \quad \text{o} \\ \text{o} \, \text{i} \quad \text{o} \quad \text{o} \qquad \text{i} \qquad \text{o} \quad \text{u} \, \text{o} \quad \text{o} \quad \text{o} \, \text{i} \, \text{i} \\ \text{o} \quad \text{k} \, \text{i} \quad \text{i} \, \text{o} \quad \text{o} \, \text{u} \qquad \text{o} \quad \text{u} \quad \text{i} \quad \text{o} \quad \text{o} \qquad \text{u} \, \text{o} \quad \text{o} \\ \text{o} \, \text{u} \quad \text{o} \, \text{i} \quad 2 \qquad \text{i} \qquad \text{o} \, \text{i} \, \text{i} \quad \text{o} \quad \text{i} \qquad \text{i} \\ \text{i} \, \text{i} \qquad \text{o} \, \text{i} \quad \text{i} \qquad \text{u} \quad \text{i} \quad \text{i} \qquad \text{u} \, \text{o} \\ \text{o} \, \text{i} \quad \text{o} \, \text{o} \, \text{i} \, \text{i} \, \text{o} \quad \text{o} \qquad \text{i} \, \text{i} \quad \text{u} \, \text{o} \quad \text{o} \quad \text{o} \, \text{u} \quad F \quad \text{i} \\ \text{o} \qquad \text{i} \quad \text{u} \quad \text{o} \quad \text{i} \quad x^\omega \quad F \quad \text{x} \, \text{i} \quad \text{o} \\ \text{u} \, \text{o} \quad \text{o} \, \text{i} \quad \text{o} \, \text{o} \, \text{i} \, \text{i} \, \text{o} \quad \text{o} \, \text{i} \quad \text{u} \qquad \text{i} \, \text{i} \, \text{o} \\ \text{i} \, \text{i} \quad \text{o} \qquad \text{i} \, \text{i} \, \text{o} \quad \ddot{\text{u}} \, \text{i} \, \text{u} \, \text{o} \quad \text{o} \end{array}$$

$$\textbf{Theorem 3.} \quad \begin{array}{c} f \quad LTL \quad X \quad u \quad Bu \, f \quad \ddot{u} \\ u \quad \sigma \quad x_0 \, x_1 \, x_2 \dots \quad v \quad AP \quad true, false \\ \sigma \quad f \qquad Bu \, f \end{array}$$

$$\rho \quad F_0^{(X_0,A_0)} \, F_1^{(X_1,A_1)} \, F_2^{(X_2,A_2)} \, \dots$$

$$\sigma \, \sigma \quad x_{i_0} \, x_{i_1} \, x_{i_2} \dots \, u$$

$$\begin{array}{l} u \\ - \quad \rho \quad \sigma \\ - \quad k < \sigma \quad - \quad i_k < i_{k+1} \end{array}$$

igning

n n ing

- $k < \sigma \quad X_k \ x_{i_k}$
- $k < \sigma \quad - , \ i \ i_k \quad i < i_{k+1} \quad x_i \ AP_k \quad x_{i_k} \ AP_k$
- $k < \sigma \quad - \quad x_{i_k} \ AP_k \quad x_{i_{k+1}} \ AP_k$
- $\sigma <$
- $i > i_{\sigma-1} \quad x_i \ AP_{\sigma-1} \quad x_{i_{\sigma-1}} \ AP_{\sigma-1}$
- $x_{i_{\sigma-1}} \ \omega \quad F_{\sigma-1}$
- $\sigma \quad a \quad Acc, \ i \ 0, \ j \ i \ a \ A_j$

$AP_i \quad Domain \ F_i$

$\sigma \ x_0 \ x_1 \ x_2 \dots \quad i \ i \ o \quad i \ i \ o \ u \ f$

$i \quad i \ u \ io \ o \quad i \ o \ Domain \ f \quad i \quad \rho$

$i \ Bu \ f \quad o \ \sigma \ i \ u \quad o \ i \ 2 \ o \quad o$

$Domain \ f \quad f \ true \ \rho \ i \ u \ o \ i \ i \quad f$

$\sigma \ x_0 \ \rho \ \sigma \ u \quad o \ i \ 2$

$Domain \ f \quad o \ i \quad u \quad o \ \sigma \quad x_{i_0} \ x_{i_1} \ x_{i_2} \dots$

$o \ i \ o \ o \ 2 \quad u \ \sigma \ f \quad i \quad i \quad i \quad i$

$o \quad i \ i \quad f \ i \ Bu \ f$

$\rho \quad F_0 \quad^{(X_0, A_0)} \quad F_1 \quad^{(X_1, A_1)} \quad F_2 \quad^{(X_2, A_2)} \quad \dots$

$u \quad j \ X_j \ x_{i_j} \quad a \ Acc, \ j \ 0, \ k \ j \ a \ A_k \quad k$

$u \quad o \ \sigma \ j \ o \ \sigma \quad i \ o \ u \ F_j \quad o \ i$

$o \ o \ i$

$j \ Domain \ F_j \quad Domain \ f \quad x_{i_j} \ Domain(f) \quad x_{i_{j+1}} \ Domain(f) \quad \sigma$

$\rho \quad ooki \quad o$

$2 \quad n \ u$

- $j \ n \ Domain \ F_j \quad Domain \ f$
- $j < n \quad x_{i_j} \ Domain(f) \quad x_{i_{j+1}} \ Domain(f)$
- $j > n \quad x_{i_j} \ Domain(f) \quad x_{i_n} \ Domain(f)$

$i \quad F_0 \dots F_n \quad o \quad x_{i_0} \dots x_{i_n} \quad ooki$

$o \quad o \quad k \quad u \quad o \quad \sigma \quad n \quad i \quad u \quad i$

$ui \quad o \ x_{i_n} \ \omega \quad i \quad o \ Domain \ F_n \quad i \quad o \quad u$

F_n

$i \quad n \ u$

- $j < n \quad Domain \ F_j \quad Domain \ f$
- $j < n \quad x_{i_j} \ Domain(f) \quad x_{i_{j+1}} \ Domain(f)$
- $Domain \ F_n \quad Domain \ f$

$o \quad u \ o \quad o \quad u \ io \quad o \quad Bu \ F_n \ i \ u \ o \ Bu \ f \ i$

$F_n \quad i \ i \ i \quad ki \quad i \ u \ io \quad o \quad o \ o \ u$

$F_n \quad u \quad o \ \sigma \ n \quad o \quad i \quad \rho \ i \ Bu \ f \quad o \quad F_n$

$o \ \sigma \ o \ \sigma \ n \quad i \ u \quad o \quad i \ 2 \quad o$

$F_0 \dots F_n \ \rho \quad o \quad x_{i_0} \dots x_{i_n} \ \sigma \quad ooki \quad o$

$$\begin{array}{cccccccccccccccc}
& & n & i & & & i & n & i & & n & & n & i & & n \\
& & \sigma & x_0 & x_1 & x_2 \dots & i & i & o & & \rho & & \sigma \\
& & o & i & u & o & i & 2 \\
& & \rho & i & i & x_{i_{\sigma-1}} & \omega & F_{\sigma-1} & u & u & o & \sigma_{\sigma-1} \\
i & u & i & u & i & o & x_{i_{\sigma-1}} & \omega & i & o & Domain & F_{\sigma-1} & i & o \\
& i & F_{\sigma-1} \\
Domain & F_n & \rho & i & i & i & i & i & n & u & j > n \\
\sigma_n & o & i & o & u & F_n & u & o & \sigma_{i_n} & i & u & i & u & i & o \\
& o & o & o & o & o & i & u & i & o & o
\end{array}$$

$$\begin{array}{cccccccccccccccc}
& & j & \sigma_{i_{j+1}} & F_{j+1} & \sigma_{i_j} & F_j \\
& i & o & i & u & i & \sigma_{i_{j+1}} & F_{j+1} & o & x_{i_j} & \sigma_{i_{j+1}} \\
& i & F_j & i & i & o & F_j & (X_j, A_j) & F_{j+1} & o & i & i \\
o & \sigma_{i_{j+1}} & F_{j+1} & i & i & i & o & x_{i_j} & \sigma_{i_{j+1}} & F_j & x_{i_j} & \sigma_{i_{j+1}} & i \\
u & i & u & i & o & \sigma_{i_j} & i & o & Domain & F_j
\end{array}$$

$$\begin{array}{cccccccccccccccc}
& & i & o & o & o & x^\omega & f & i & i & i \\
& o & i & o & o & u & o & o & o & o & U & o \\
o & o & i & i & o & o & o & o & o & i & o & i & i \\
i & i & i & o & x^\omega & f & x^\omega & f & x^\omega & f & g & x^\omega & f & x^\omega & g
\end{array}$$

Proposition 1. $f \ g \ LTL \ X \ u \ v \ x \ AP$
true, false

$$\begin{array}{cccccccccccccccc}
& & x^\omega & fUg & x^\omega & g \\
& o & u & o & o & i & i & i & i & o & o & i & i & o & o & fUg \\
o & i & i & o & i & u & o & u & o & x^\omega & u \\
& x^\omega & fUg & x^\omega & g & x^\omega & f & x^\omega & g \\
& i & i & i & i & o & u & o & i & o & u & o & o
\end{array}$$

$$\begin{array}{cccccccccccccccc}
& & i & o & u & o & o & i & i & o & dead & o & o & ki & o \\
dead & m & m & i & i & o & i & i & i & o & o & i & i & o \\
i & u & o & o & o & i & i & o & dead & o & i & i & u & i \\
u & o & o & u & o & o & o & o & i & o & o & i & i & o \\
o
\end{array}$$

Definition 5 (Legal word). $\sigma \langle x_0, d_0 \rangle \langle x_1, d_1 \rangle \dots$
 $v \ AP \ true, false \ true, false \ \sigma$

$$\begin{array}{cccccccccccccccc}
& & i, \ j > i & d_i & d_j & x_j & x_i \\
& o & LTL^d & i & o & o & i & o & o & i & o & o & i & o & o & i \\
io & AP & dead & o & i & o & LTL^d & X & o & u \\
i & i & u & o & o & i & o & o & u & i & o & i
\end{array}$$

igning

n n ing

u i o i io o o i o o i io i i o u o
u o o i ou ki i o ou o o i io *dead*
oi o o u oi o o o o i
io *dead* i o o i o u i o i io oi 2 u
o i i o i o o i u o o
u o o i o o i io i o o i io
o u o o o i io *dead* i i u o i o i
i io o u o o o u *F* o i u o

– *dead* oi 2 $\langle x, false \rangle^\omega$ *F*
– i *dead* oi 2 $\langle x, true \rangle^\omega$ *F*
– o i i i u u *dead* oi 2 $\langle x, false \rangle$ $\langle x, true \rangle^\omega$ *F*

x i o u o o i o o i io oi u
i i o i i o i io ü i u o o

Theorem 4. $f \quad LTL^d \quad X \quad u \quad Bu \quad f \quad \ddot{u}$
 $u \quad \sigma \quad \langle x_0, d_0 \rangle \quad \langle x_1, d_1 \rangle \quad \langle x_2, d_2 \rangle \dots \quad v$
 $AP \quad true, false \quad true, false \quad \sigma \quad f$
 $Bu \quad f$

$\rho \quad F_0^{(X_0, A_0)} \quad F_1^{(X_1, A_1)} \quad F_2^{(X_2, A_2)} \dots$

$\sigma \quad \sigma \quad \langle x_{i_0}, d_{i_0} \rangle \quad \langle x_{i_1}, d_{i_1} \rangle \quad \langle x_{i_2}, d_{i_2} \rangle \dots \quad u$

u

– $k < \sigma \quad i_k < i_{k+1}$
– $k \quad X_k \quad \langle x_{i_k}, d_{i_k} \rangle$
– $k < \sigma \quad -, \quad i \quad i_k \quad i < i_{k+1} \quad x_i \quad AP_k \quad x_{i_k} \quad AP_k$
– $k < \sigma \quad - \quad x_{i_k} \quad AP_k \quad x_{i_{k+1}} \quad AP_k$
 $\sigma < \quad i > i_{\sigma-1} \quad x_i \quad AP_{\sigma-1} \quad x_{i_{\sigma-1}} \quad AP_{\sigma-1}$
 $i \quad i_{\sigma-1} \quad d_i \quad \langle x_{i_{\sigma-1}}, false \rangle^\omega \quad F_{\sigma-1}$
 $d_{i_{\sigma-1}} \quad \langle x_{i_{\sigma-1}}, true \rangle^\omega \quad F_{\sigma-1}$
 $d_{i_{\sigma-1}} \quad i > i_{\sigma-1} \quad d_i \quad \langle x_{i_{\sigma-1}}, false \rangle \quad \langle x_{i_{\sigma-1}}, true \rangle^\omega \quad F_{\sigma-1}$
 σ
– $a \quad Acc, \quad i \quad 0, \quad j \quad i \quad a \quad A_j$

$AP_i \quad Domain \quad F_i \quad dead$

oo o i o i iou i o oo
o o o oo i u io i o o i
o *Domain f* *dead*

i o o o $\langle x, false \rangle^\omega$ *f* $\langle x, true \rangle^\omega$ *f*
 $\langle x, false \rangle$ $\langle x, true \rangle^\omega$ *f* i i o i o
o u u i o o i io 2 i io o oo o o

n i i n i n ni i n

Proposition 2. $f \ g \quad LTL^d \ X \quad u$
 $x \ AP \quad true, false$

$$\begin{aligned} \langle x, false \rangle^\omega \quad fUg \quad \langle x, false \rangle^\omega \quad g \\ \langle x, true \rangle^\omega \quad fUg \quad \langle x, true \rangle^\omega \quad g \\ \langle x, false \rangle \quad \langle x, true \rangle^\omega \quad fUg \quad \langle x, false \rangle \quad \langle x, true \rangle^\omega \quad g \\ \langle x, false \rangle \quad \langle x, true \rangle^\omega \quad f \quad \langle x, true \rangle^\omega \quad g \end{aligned}$$

o o u i u i io o o o i io
i o u i u o i io o fUg o i i o i-
u o u o $\langle x, false \rangle \quad \langle x, true \rangle^\omega$ o i o o i
o u

$$\begin{aligned} \langle x, false \rangle \quad \langle x, true \rangle^\omega \quad fUg \quad \langle x, false \rangle \quad \langle x, true \rangle^\omega \quad g \\ \langle x, false \rangle \quad \langle x, true \rangle^\omega \quad f \quad \langle x, true \rangle^\omega \quad g \\ \langle x, false \rangle \quad \langle x, true \rangle^\omega \quad f \quad \langle x, true \rangle^\omega \quad f \quad \langle x, true \rangle^\omega \quad g \end{aligned}$$

i i i i o u o i o u o o

4 Verification of $LTL \setminus X$ Formula with Unfolding Graphs

i io o o u o i i
u o i io o $LTL \setminus X$ o u i i i fl i io
o u o i o u i o i o
i io ü i u o o o o i o o u o u o
i i o u o u i i
i u o i io o o - k o i o
u o i o i i u o i o o
i ki u u o i i u i i u
o i u o i o u io o o i u o i i o -
i u ii u o o o i i
u o i u i i u i

Definition 6 (Partial size rule unfolding). $u \ u$
 $N \quad u \quad \beta, Bridge, \quad u, \phi \quad u \quad \beta, Bridge \quad u$
 $N \quad \phi \quad u \quad E \beta \quad Bridge$

$$u \quad u \quad h \beta \quad Cut \quad e \quad h \beta \quad Cut \quad \phi \quad e \quad \phi \quad e \quad < \quad e$$

i i o i u o i o o i o o
i i u o o o i o o ii
io i u i o o

Definition 7 (Unfolding graph).

$$\begin{aligned}
 & u \quad N \quad G, \quad , g_0 \\
 & - \quad g \quad G \quad g \quad u \quad u \quad N, h \quad g \quad Min \quad g \quad N \\
 & \quad \quad \quad h \quad g \quad Min \quad g \quad h \quad g \quad Min \quad g \quad Reach \quad N \\
 & - \quad g \quad G, \quad e \quad Bridge \quad g, \quad g \quad G \quad g^{-e} \quad g \quad h \quad g \quad Cut \quad e \quad h \quad g \quad Min \quad g \\
 & - \quad h \quad g_0 \quad Min \quad g_0 \quad m_0 \\
 & - \quad g \quad G \quad G, \quad , g_0 \quad g_0 \quad g \\
 & \quad \quad \quad i \quad i \quad u \quad u \quad i \quad i \quad io \quad o \quad u \quad o \quad i \quad o \\
 & o \quad \quad \quad io \quad o \quad o \quad i \quad io \quad i \quad u \quad io \quad u \quad o \quad i \\
 & \quad \quad \quad i \quad io \quad o \quad io \quad o \quad u \quad o \quad o \quad i \\
 & io
 \end{aligned}$$

Definition 8 (Partial inclusion rule unfolding).

$$\begin{aligned}
 & u \quad u \quad \beta, Bridge, \quad u \quad , \phi \quad u \quad u \quad u \\
 & e \quad u \quad \phi \quad e \quad e \\
 & \quad \quad \quad i \quad io \quad o \quad i \quad io \quad o \quad u \quad o \quad i \\
 & \quad \quad \quad i \quad io \quad \ddot{u} \quad i \quad u \quad o \quad o \quad ki \quad i \quad o \quad ou \quad u \quad i \quad i- \\
 & io \quad o \quad u \quad o \quad i \quad o \quad o \quad o \quad i \quad u \quad o \quad o \quad i \quad o \quad o \quad i \quad io \\
 & i \quad o \quad u \quad i \quad io \quad i \quad o \quad i \quad o \quad u \quad i \quad o \quad o \\
 & \quad \quad \quad o \quad u \quad o \quad o \quad o \quad o \quad i \quad i \quad o \quad u \quad i \quad io \quad i \\
 & \quad \quad \quad u \quad o \quad o \quad o \quad i \quad u \quad o \quad u \quad o \quad o \quad i \quad i \quad i- \\
 & i \quad ki \quad o \quad u \quad u \quad o \quad i \quad o \quad u \quad i \quad o \quad o \quad i \quad io \\
 & \quad \quad \quad i \quad i \quad io \quad o \quad u \quad o
 \end{aligned}$$

Definition 9 (Synchronization).

$$\begin{aligned}
 & \langle N, m_0 \rangle \quad \langle G, \quad , \quad \rangle \\
 & g_0 \rangle \quad u \quad \langle N, m_0 \rangle \quad f \quad LTL \quad X \quad u \quad Bu \quad f \\
 & \langle Q, Acc, \quad , \quad f \rangle \quad \ddot{u} \quad u \quad \ddot{u} \quad u \\
 & u \quad \langle G \quad Q, Acc, \quad , \langle g_0, \quad f \rangle \rangle \\
 & - \quad G \quad Q \quad PowerSet \quad Acc \quad G \quad Q \\
 & - \quad \langle g, F \rangle^{-A} \quad \langle g, F \rangle \\
 & \quad \quad \quad \Phi_{Domain(F)} \quad m_0 \quad g \quad \Phi_{Domain(F)} \quad m_0 \quad g \quad e \quad g^{-e} \quad g \quad A \\
 & \quad \quad \quad F \quad F \\
 & \quad \quad \quad \Phi_{Domain(F)} \quad m_0 \quad g \quad \Phi_{Domain(F)} \quad m_0 \quad g \quad e \quad g^{-e} \quad g \quad X \\
 & \quad \quad \quad F^{(X,A)} \quad F \quad X \quad m_0 \quad g \\
 & \quad \quad \quad o \quad u \quad io \quad o \quad o \quad i \quad o \quad u \quad o \quad i \quad i \quad i \quad ki \quad o \\
 & \quad \quad \quad i \quad u \quad o \quad i \quad o \quad o \quad i \quad o \quad o \quad i \quad o \quad i \quad i \quad u \quad o \\
 & o \quad \quad \quad i \quad io \quad i \quad i \quad i \quad ki \quad i \\
 & \quad \quad \quad o \quad i \quad o \quad i \quad o \quad o \quad i \quad io \quad i \quad o \quad o \\
 & \quad \quad \quad i \quad o \quad u \quad i \quad o - \quad i \quad o \quad i \quad u \quad i \quad o \quad i \quad o- \\
 & u \quad \quad \quad o \quad io \quad o \quad o \quad i \quad io \quad o \quad o \quad o \quad i \quad io \quad i \quad io \quad i \\
 & ou \quad o \quad i \quad o \quad i \quad u \quad o \quad o \quad o \quad i \quad io \quad o \quad i \quad \langle g, F \rangle \quad o \\
 & \quad \quad \quad o \quad i \quad o \quad u \quad o \quad i \quad io \quad o \quad o \quad o \quad i \quad io \\
 & o \quad F \quad o \quad u \quad i \quad i \quad u \quad o \quad i \quad g \quad i \quad io \quad o \quad i
 \end{aligned}$$

Definition 10 (Observed transition). $\langle N, m_0 \rangle$ p
 $v \text{ Reach } N, m_0$ v p
 $Obs \ p \ u$

$t \ T \ m, m \text{ Reach } N, m_0, m \text{---}^t \ m \ p \ m \ p \ m \ Obs \ p$

i io i i i o o u io k i o ou
o i o o i io o o i o
i u io u i u i o o i i i
io o i i i ki i i i i
o u

Definition 11 (Compatibility). $\langle N, m_0 \rangle$ $\langle G, _, g_0 \rangle$
 $u \langle N, m_0 \rangle \ f \ LTL \ X \ u \ Bu \ f \ \langle Q, Acc, _, _ \rangle$
 $f \rangle \ \ddot{u} \ u \ u \ \langle G \ Q, Acc, _, _ \rangle$
 $\langle g_0, f \rangle \rangle \ u \ \ddot{u} \ u$
 $\langle g, F \rangle \ G \ Q$

– $e \ E \ g \ Bridge \ g, \ p \ Domain \ F \ h \ e \ / \ Obs \ p$
– $\Phi_{Domain(F)} \ m_0 \ g \ \omega \ F \ g \ u \ u \ u$
i i o i io o i o i -
o i o u

Lemma 1. $\langle g, F \rangle$ $u \ \langle G$
 $Q, Acc, _, \langle g_0, f \rangle \rangle \ m \ m$

$m \ h \ g \ Min \ g \ \pi \ m \ m$
 $v \ F$
 $u \ \langle g, F \rangle$

$\nu \ \langle g, F \rangle \ \langle g_1, F \rangle \ \langle g_n, F \rangle$

$u \ m \ Reach \ g_n$
 $m \ Reach \ g \ \Phi_{AP} \ m \ Domain(F) \ \Phi_{AP} \ m \ Domain(F) \ m \ m$
 $N \ u \ \langle g, F \rangle$

$\nu \ \langle g, F \rangle \ \langle g_1, F \rangle \ \langle g_n, F \rangle$

$u \ m \ Reach \ g_n \ v \ \nu \ u \ \langle g, F \rangle$
 $u \ m \ g \ u \ u \ m$
 $g \ u$
 $m \ Reach \ g \ \Phi_{AP} \ m \ Domain(F) \ \Phi_{AP} \ m \ Domain(F) \ m \ m$
 $N \ v \ F \ \overset{(X,A)}{_} \ F \ F \ Bu \ f \ u$
 $X \ \Phi_{AP} \ m \ u \ \langle g, F \rangle$

$\nu \ \langle g, F \rangle \ ^A \ \langle g, F \rangle \ \langle g_n, F \rangle$

igning

n n ing

i i o i u io o i o o π
 o π i u o i o iou m m i
 u o o i o o i
 - o π i i g C g C Cutout Π C π
 i u o o $\langle g, F \rangle$
 - o π u u o e u , C π Π e
 Π C i u io o o o π
 Π Φ e . π Π e i i π i i i
 o π o i o o i io
 - o π u i e , C π Π e
 Π C i u io o o o π
 π Π e π i π h g Min g
 i i i h g Min g g^{-e} g o π o -
 i o o i io i i o i o
 i io $\langle g, F \rangle$ $\langle g, F \rangle$ u o i u io
 2 o o o i i u o o
 o o i o π i g i m
 i io π i i io m m π i i g m i i
 Reach g i o u io i o o m i π
 u i o u o i i o i o i io m m
 o o u o o i o i o
 u o o i o o i o o i io
 o o o i o i i o u f
 i i o i o i o u o o i o i i
 o i i f o i o u i
 i i io \ddot{u} i u o o o i o u i
 o o o oi 2 i o i io
 \ddot{u} i u o o i i o i o u o i
 i o i io u i i i i i o i
 o o i io o i i oi o
 o

Theorem 5 (Checking property). $\langle N, m_0 \rangle$ $\langle G, , g_0 \rangle$
 $\langle Q, Acc, , f \rangle$ $\langle N, m_0 \rangle$ f LTL X u Bu f
 \ddot{u} u $\langle G$ Q, Acc, , $\langle g_0, f \rangle \rangle$
 \ddot{u} u σ $\langle N, m_0 \rangle$ u f
 $\langle G$ Q, Acc, , $\langle g_0, f \rangle \rangle$
 ν $\langle g_0, f \rangle \xrightarrow{A_0} \langle g_1, F_1 \rangle \xrightarrow{A_1} \langle g_2, F_2 \rangle \xrightarrow{A_2} \dots$

- $\nu <$ $\Phi_{Domain(F_{\nu-1})}$ m_0 $g_{\nu-1}$ ω $F_{\nu-1}$ $g_{\nu-1}$
 u

n i i n i n ni i n

$$- \nu \quad \overset{u}{i} \quad \overset{j > i}{j} \quad \overset{\omega}{\Phi_{Domain(F_i)} m_0 g_i} \quad \overset{\Phi_{Domain(F_i)} m_0 g_j}{F_i}$$

$$- \nu \quad \overset{u}{a} \quad \overset{u}{Acc}, i \quad 0, j \quad i \quad a \quad A_j$$

$$\begin{matrix} \sigma & \Phi_{AP} m_0 & \Phi_{AP} m_1 & \Phi_{AP} m_2 \dots & i & i & o \\ \langle N, m_0 \rangle & & & & & & \\ o & o & i & u & o & \sigma & \Phi_{AP} m_0 \\ \Phi_{AP} m_{i_1} & \Phi_{AP} m_{i_2} \dots o & \sigma & i & Bu & f \end{matrix}$$

$$\rho \quad F_0^{(X_0,A_0)} \quad F_1^{(X_1,A_1)} \quad F_2^{(X_2,A_2)} \dots$$

$$\begin{matrix} i & u & o & i & 2 & o & o \\ & & o & i & io & m & m & o & i & i \\ \nu & & & & & & & & & \end{matrix}$$

$$\nu \quad \langle g_0, F_0 \rangle \quad \dots \langle g_0, F_0 \rangle^{A_0} \langle g_1, F_1 \rangle \quad \dots \langle g_1, F_1 \rangle^{A_1} \langle g_2, F_2 \rangle \dots$$

$$o \quad u \quad o \quad i \quad i \quad o \quad o \quad i$$

$$\begin{matrix} - i & \rho & i & i & i & o & o & o & \nu & i & i & i & i \\ & & & & i & o & i & io & o & & o & & \\ - i & \rho & i & i & i & o & 2 & o & o & \sigma & i & i & \nu & i & i & i & i \\ & o & o & i & & i & ki & & i & & i & o & i & io \\ & o & & o & & & & & & & & & & & & \\ - i & \rho & i & i & i & o & 2 & o & o & \sigma & i & i & i & \nu & ou & i \\ & o & i & i & \nu & i & i & & o & u & o & i & u & o & o & 2 \\ i & \nu & i & i & i & i & i & i & i & u & u & o & i & & u \\ & u & i & & o & & o & & i & o & i & io & i \\ & & & i & o & i & io & o & \rho & i & o & u & u & o & o \end{matrix}$$

$$\begin{matrix} i & o & oo & i & o & iou & u & o & k & o \\ i & io & \langle g, F \rangle^A & \langle g, F \rangle & i & o & i & o & u & o & o & u \\ o & & o & h & g & Min & g & o & h & g & Min & g & i & io & F & \overset{A,X}{-} \\ F & i & i & o & i & io & o & o & i & o & o & i & io & o & F & h & g & Min & g \\ & h & g & Min & g & o & o & u & u & o & & & & & & & & \\ o & o & u & u & o & o & i & i & o & i & o & u & o & 2 \\ o & & o & u & i & i & u & i & u & i & & o & & & & & & \\ o & i & i & ki & o & i & i & u & i & o & o & i \\ u & o & & & & & & & & & & & & & & \end{matrix}$$

$$\begin{matrix} o & i & o & i & i & io & o & i & o \\ & o & i & o & i & o & u & o & o & i & i & i & - \\ o & o & ii & o & o & i & ui & o & i \\ i & u & o & o & i & o & o & i & io & o & u & o & o & o & - \\ u & i & i & i & o & o & - & - & i \end{matrix}$$

o i o u i i io i o i io i o o
 o u o o o i o o o i u o o i
 o o i io i o i i o o
 o o o i o i i o i io u
 o i i 2 i i o - -
 ki o io o i i i o - -fl -
 o i o - - i i io i o i io
 i o o u o o o u io i o i
 o u o u o i i i o o
 o ki u o o o u F o i i
 o i u o i i u io o i u u o u i i i
 u i i o ki i i i i F
 i i i u io u i u o i io i
 o F o i i u io u u o i o i o
 io o o oo u o i
 u o o o i o o
 i u o i u i o i i
 i o o o u o ki i u o i
 o i o o o i u o o i o o i io o i o
 F o i o i o u o o
 o o i o u o o i io ü i u o o
 o i i u ki
 o i io o o i io i
 o o u io o io o o 2

5 Verification of $LTL^d \setminus X$ Formula with Unfolding Graphs

i io o o u o i u o i io
 o $LTL \setminus X$ o u u i o i o o i io dead i o
 u o i i io i i i o o o i i
 o o i io o o o o i
 i u i i o $LTL^d \setminus X$ o u
 o o i io dead i o k i o ou o i i i i
 o o i io o i io ü i u o o u o i

Definition 12 (Dead synchronization). $\langle N, m_0 \rangle$ $\langle G,$
 $\langle g_0 \rangle$ u $\langle N, m_0 \rangle$ f $LTL^d \setminus X$ u $Bu f$
 $\langle Q, Acc, \cdot, f \rangle$ \ddot{u} u \ddot{u} u $\langle G$
 u \ddot{u} u $\langle G$
 $Q, Acc, \cdot, \langle g_0, f \rangle \rangle$

– $G \ Q \ PowerSet \ Acc \ G \ Q$

– $\langle g, F \rangle \xrightarrow{A} \langle g, F \rangle$

$\Phi_{Domain(F)} \text{ dead } m_0 \ g \ \Phi_{Domain(F)} \text{ dead } m_0 \ g$
 $e \ g \xrightarrow{e} g \ A \ F \ F$

n i i n i n n i i n

$$\Phi_{Domain(F)} \text{ dead } m_0 g \quad \Phi_{Domain(F)} \text{ dead } m_0 g$$

$$e \quad g \xrightarrow{e} g \quad X \quad F \xrightarrow{(X,A)} F \quad X \quad m_0 g$$

i io i i i o o u io i o o i
o i o i io o o i io o
i u io u i i o o u i i
i i u o o i o o i io o i i i ki
o i o o i io dead i i o u o i o
o

Definition 13 (Dead compatibility).

$$\langle N, m_0 \rangle \quad \langle G, \text{ } \rangle$$

$$\langle Q, Acc, \text{ } , f \text{ } \rangle \quad \langle N, m_0 \rangle f \quad LTL^d \quad X \quad u \quad Bu f$$

$$Q, Acc, \text{ } , \langle g_0, f \text{ } \rangle \quad \ddot{u} \quad u \quad u \quad \langle G$$

$$\langle g, F \rangle \quad G \quad Q \quad \ddot{u} \quad u$$

- $e \quad E \quad g \quad Bridge \quad g \text{ } , p \quad Domain \quad F \quad h \quad e \quad / \quad Obs \quad p$
- $\langle \Phi_{Domain(F)} \text{ dead } m_0 g \text{ } , false \rangle^\omega \quad F \quad g \quad u \quad u$

o o o i o i i LTL^d X o u
i o o i o i o u o o
k o o o o i o u
i o i i i u u
o i i u i i u - i
o i i i i u i i i i ou
u i o o i i io i i
oi 2 o o o o o i io dead
u i i o i i o oi 2 i i i o
oi
o o i oi o o i u
o i o o u o u io o i u o i o o i
u i o u i io i i i o
o o o o i ki
o o o i i i u u oi o o
o o i i i io u i o i ki
i o u o i i u i i i ki -
i o u o i u o i i i o o u
u o i u o o

Theorem 6 (Dead checking property).

$$\langle N, m_0 \rangle \quad \langle G, \text{ } \rangle$$

$$\langle Q, Acc, \text{ } , f \text{ } \rangle \quad \langle N, m_0 \rangle f \quad LTL^d \quad X \quad u \quad Bu f$$

$$\ddot{u} \quad u \quad \langle G \quad Q, Acc, \text{ } , \langle g_0, f \text{ } \rangle \rangle$$

$$u \quad u \quad \sigma \quad \langle N, m_0 \rangle \quad u \quad f$$

$$\langle G \quad Q, Acc, \text{ } , \langle g_0, f \text{ } \rangle \rangle$$

igning

n n ing

$$\nu \quad \langle g_0, f \rangle \xrightarrow{A_0} \langle g_1, F_1 \rangle \xrightarrow{A_1} \langle g_2, F_2 \rangle \xrightarrow{A_2} \dots$$

$$\begin{aligned} \nu < & \langle \Phi_{AP_{\nu-1}} m_0 g_{\nu-1}, false \rangle^\omega \quad F_{\nu-1} \quad g_{\nu-1} \\ u & \\ \nu & \\ i & \quad j > i \quad \Phi_{AP_i} m_0 g_i \quad \Phi_{AP_i} m_0 g_j \quad \langle \Phi_{AP_i} m_0 g_i, false \rangle^\omega \quad F_i \\ \nu & \quad a \quad Acc, i \quad 0, j \quad i \quad a \quad A_j \\ \nu < & \langle \Phi_{AP_{\nu-1}} m_0 g_{\nu-1}, true \rangle^\omega \quad F_{\nu-1} \quad \Phi_{AP_{\nu-2}} m_0 g_{\nu-1} \\ \Phi_{AP_{\nu-2}} m_0 g_{\nu-2} & \quad e \quad Bridge g_{\nu-2} \quad g_{\nu-2} \xrightarrow{e} g_{\nu-1} \quad C \\ Conf g_{\nu-2} & \quad C \quad Bridge g_{\nu-2} \quad e \quad C \quad Cutoff g_{\nu-2} \\ Dead h Cut C & \\ \nu < & \langle \Phi_{AP_{\nu-1}} m_0 g_{\nu-1}, false \rangle \quad \langle \Phi_{AP_{\nu-1}} m_0 g_{\nu-1}, true \rangle^\omega \\ F_{\nu-1} & \quad g_{\nu-1} \\ E g_{\nu-1} & \quad \Phi_{AP_{\nu-2}} m_0 g_{\nu-1} \quad \Phi_{AP_{\nu-2}} m_0 g_{\nu-2} \\ AP_i & \quad Domain F_i \quad dead \end{aligned}$$

oo i iou i o oo o o

io i u o i io o oi
o io o o u i o u io o
o i o i o u o LTL^d X o u o i
o o i o i i i u u
o o o o ii o i
io o i o i io i o o i i i u u
o i io i o o o i i io o i-
i u i i o i o io o i-
i i u o i
o o i io o i
o i u u i i o i o i i o u io
o o u io o u o i o i i o u o i
o fli
ou o ou i io i i u o i o -
i k i i ou u o i io
o o i i o u io u o i i
u o i o fli i u o i io

6 Experimentation

u i i o i i u io o ou i u o i u o
o i o i io o i o u io u o
i u o 2 u u o i o o u
o o o i o i i i i o i i -
o i o u i u o i u u o i

n i i n i n ni i n
 o i o i o o i o io i u o i o
 i
 u u io i o o i o o u u
 u i o o o o i o o o
 i u u o i o i i o
 i

	n		n ing		
	n	g	n	g	g n i
		9			9 9 9
i i	9			9	
	9	9			

Table 1. i u

o o o i i o i i i
 o i o u o i i o
 o i u o u io o i o u o i o
 i i u i i u o i u i
 o i i io i u o o u i
 u io u o i i o io o
 o u u i o o i o o o *live* - i o
 i o i i i o u oo i u o
 u o i i u o u o o i
 o o o o i o o o *safe* -20 o - -fl i u
 o u o o oo ou i
 i oo u o i o o i iou u
 i io o i u u
 o u o i i o u o i o i o i
 u o u o i u io u o i o
 i i i i i i o o o ou o i fl i
 u o i ki o u i i o io o o i i

igning

n n ing

io i i o o u o i u io i i i
i u i o i io o i io i
i u i io oi i u o i u io u

7 Conclusion

i i o o ki o *LTL* X
o u u i u o i i u i o i u io o u o i
i io o i io o u - o - o -
-fl i io o o u i 2 i o
o
- - -fl i io o i
- i o u i o i i i o
o i o o i io i o u u o o o
- i io o i o i io i o i
i k i o i o o u
- io o i io o i 2 i ou o
- i o i io o *dead* o i o o i io
i u o i i o ou o o -
k u o i u io u o o u -
io o u u o i o o i o io u u i i
o u i o i io o i io i
o i i i i o i i i o o i u
i io ou o o o i u o
o u io ou o o i u io o u o i

References

g i i i n n n i i n
Design 99 *Formal Methods in System*
n fl i i n in gi n *Proc. of FM'99*
Lecture Notes in Computer Science g ing
g 999
g n *Proc. of Formal Description Techniques IX, Theory, Applications and*
Tools g 9 99
n i n i n i g i n
n ing n *Proc. of ICATPN'99* 9 *Lecture Notes in Computer*
Science g ing g 999
ng i n ing i n *Acta Informatica* 9
99
ing ing n n ing n *Proc. of TAPSOFT'93*
Lecture Notes in Computer Science g ing g
99

n i i n i n ni i n
 n n ing g i n n
 n i i n n *Proceedings of CONCUR'99* n in g
 ing 999
 n g n i n i n n ing
 g i n *Proc. of TACAS'96* *Lecture Notes in Computer*
Science g ing g 99
 9 i n i n fl i
 i i n in gi n *Proc. 15th Work. Protocol Specification,*
Testing, and Verification n 99 n
 i i i n n n
Lecture Notes in Computer Science ing g 99
 i n nn n i i n
 i n *Proc. 13th Int. Conf on Protocol Specification, Testing, and Verification,*
INWG/IFIP g 9 i g gi 99
 nn *Design and Validation of Computer Protocols* n i
 ng i 99
 nn n nn i n n
 n *The Spin Verification System* g i n i i
 99 n in
 n i in in n n
 n i i n n ing n *Proc. of ICATPN'96*
 9 *Lecture Notes in Computer Science* g ing g 99
 ng n in ni g n
Proceedings of the 11th International Conference on Computer Aided Verification,
Italy n in g 9 ing 999
 i n ing n ing i i n in
 i i n n n i i n *Proc. of the th Conference on Computer*
Aided Verification *Lecture Notes in Computer Science* g
 ing g 99
 n ing ing n n ing n *Proc. of*
the 9th Conference on Computer Aided Verification in
 i n g ing g 99
 i n in n in i n n n in
Theoretical Computer Science 9
 9 n n n ing ing n i
 n *Proc. on the th Conference on Computer Aided Verification* 9
Lecture Notes in Computer Science g 9 ing g 99
 n n i in i n i
 i i n i *Information Processing Letters*
 99
 i n *Graphes de Processus Arborescents pour la Vérification de Pro-*
priétés ni i i i n 99
 i n i i i n in
 gi *Journal of the Association for Computing Machinery* 9
 9
 i n g n i n n *Advances in*
Petri Nets *Lecture Notes in Computer Science* g 9
 ing g 99
 i n fl i i n i n *Proc. of the th Confer-*
ence on Computer Aided Verification 9 *Lecture Notes in Computer*
Science g 9 ing g 99

igning

n n ing

ni i n
Advances in Petri Nets

n in i n
Lecture Notes in Computer Science g

9 9 ing g 99

n ing ing n n ing n *Proc. on the* th
Conference on Computer Aided Verification in i n

ing g 99

Process Semantics of Petri Nets over Partial Algebra

..

*

..

..

..

..

{joerg.desel,gabriel.juhas,robert.lorenz}@ku-eichstaett.de

Abstract.

1 Introduction

m p p m

p p m k p p m

p p m p

p m m

m p p p

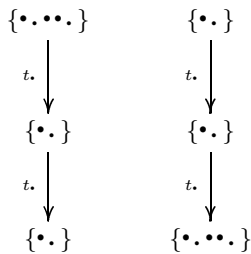
m k m p m fl m k p

*

m mp mp fl m
 p m p m
 m k — fl
 p p
 p mp p p m
 p p
 m p pp p
 p p
 p mp p m
 p k m mp
 p p m p
 p m mp p
 p m k
 p mp p
 m pp p
 p mp k
 p p
 p mp m m
 p p p
 m p
 p p m
 mp p m
 j pp m p
 p p m m
 p p p m
 p p
 p fl k
 m
 m p
 k m k
 mp m p
 p mp pp

$$\begin{array}{ccccccc}
 & & \mathbb{N} & & & & \\
 & m & & A & & & m \\
 & m & & & & & \\
 & & & & m & A & \\
 m & & & & & & \mathbb{N}^A \quad p \quad C \\
 & & m & + & & & \\
 & & & & & &
 \end{array}$$
[illegible]

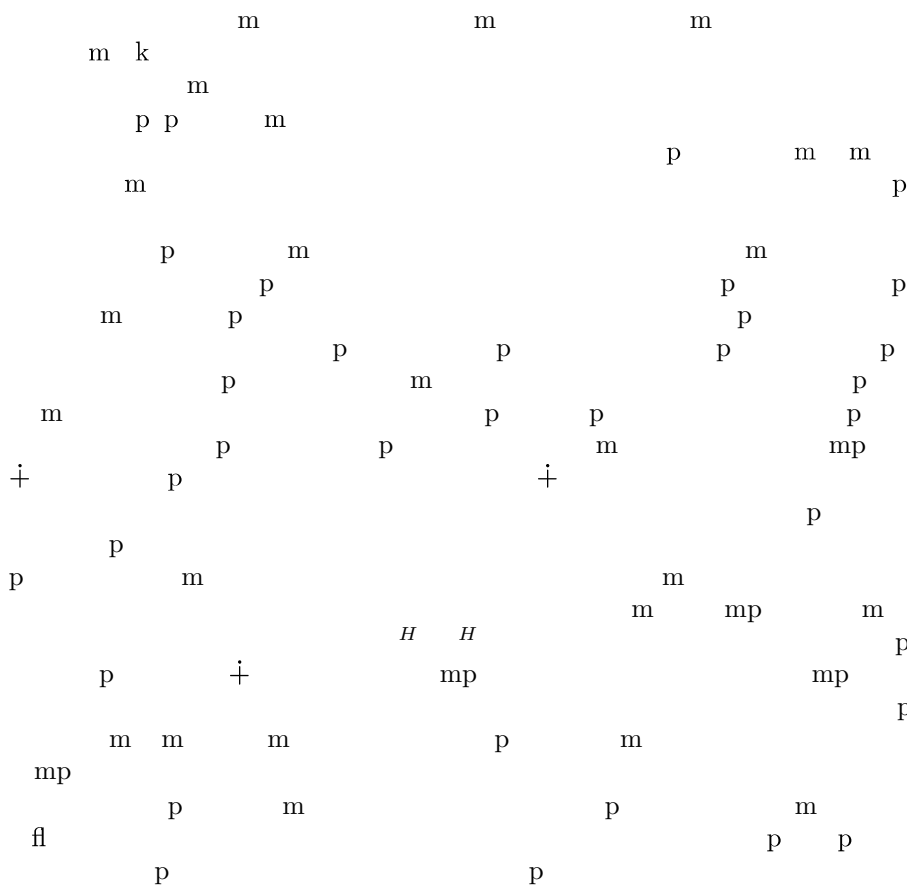
on 2.



p

p

p

 dom 

Definition 5.

$$\begin{array}{c}
\begin{array}{cc}
\dot{+} & \dot{+} \\
+ & +
\end{array} \\
= \\
\begin{array}{cc}
\dot{+} & \dot{+} \\
+ & +
\end{array}
\end{array}
\qquad
\begin{array}{cc}
H & H \\
H & \begin{array}{cc} \dot{+} & \dot{+} \\ + & + \end{array}
\end{array}$$

$y \quad y$

-

+
-
+
+=

-
=

mp
p
m

..

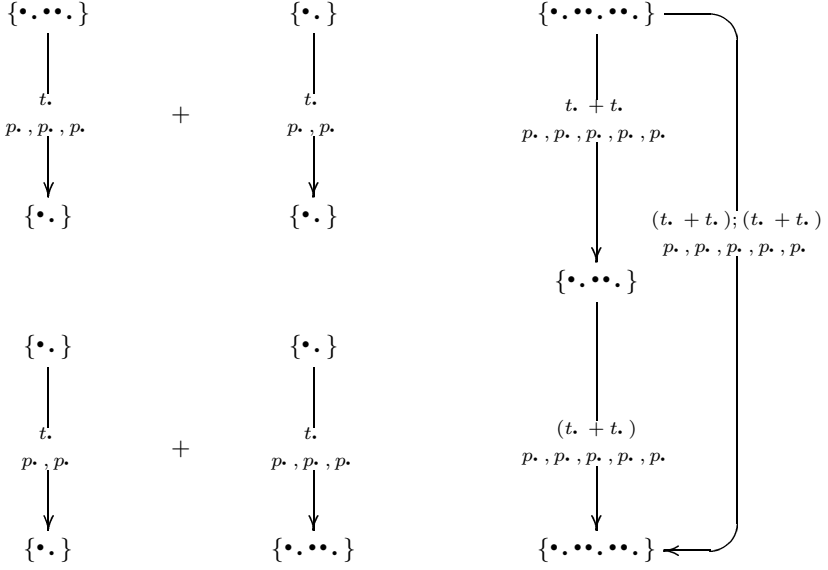


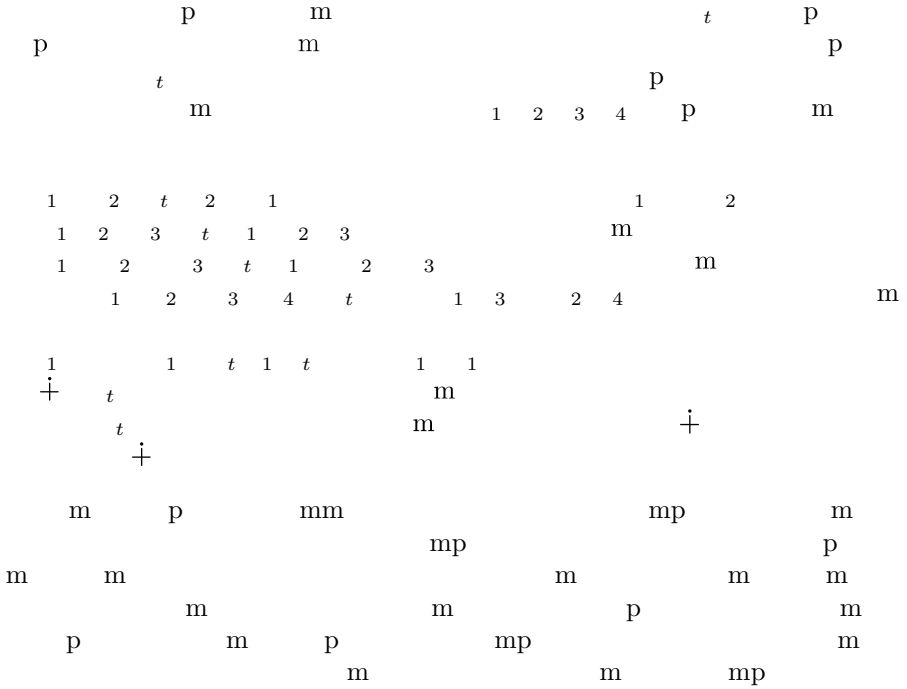
Fig. 3.

p

m

m

3.1 Equivalence of Process Terms



[illegible]

4 Elementary Nets

$$\begin{array}{ccccccc} & & m & & m & & m \\ & & m & & m & & m \\ m & k & & mp & m & & p \end{array}$$

Definition 6.
 P

$$\begin{array}{ccccccc}
 & P & & y & & & \\
 & & & P & P & & \\
 & & & & & H & \\
 & dom & & P & & & \\
 \cup_a A & & & & & & \\
 p & m & & m & & & \\
 & & H + & + & & m m & \\
 m pp & & & m + & p & & \\
 p m & & & m_H + & + & &
 \end{array}$$

Lemma 1.

Lemma 2. $\begin{matrix} & H & H \\ H & \dot{+} & \dot{+} \end{matrix}$

..

m H p p
H + +
mp m +
p p m m

Definition 7.

P y P
y j pp p
m p m m m
m p m
p

Definition 8.

y fl

y y 0
y 0
m p p m p
p p

Definition 9.

y P y
m p m
p p m m
m p
m pp p
m

Definition 10.

N N + N y N N N
N ≤ y

..

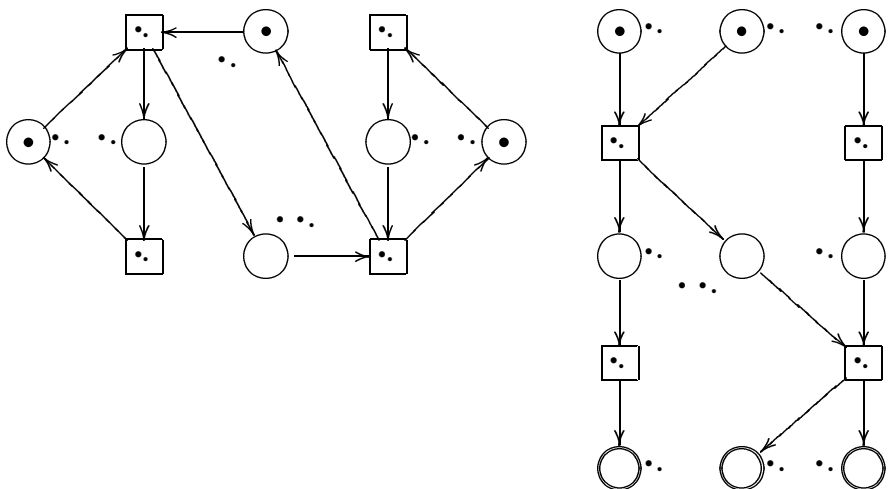
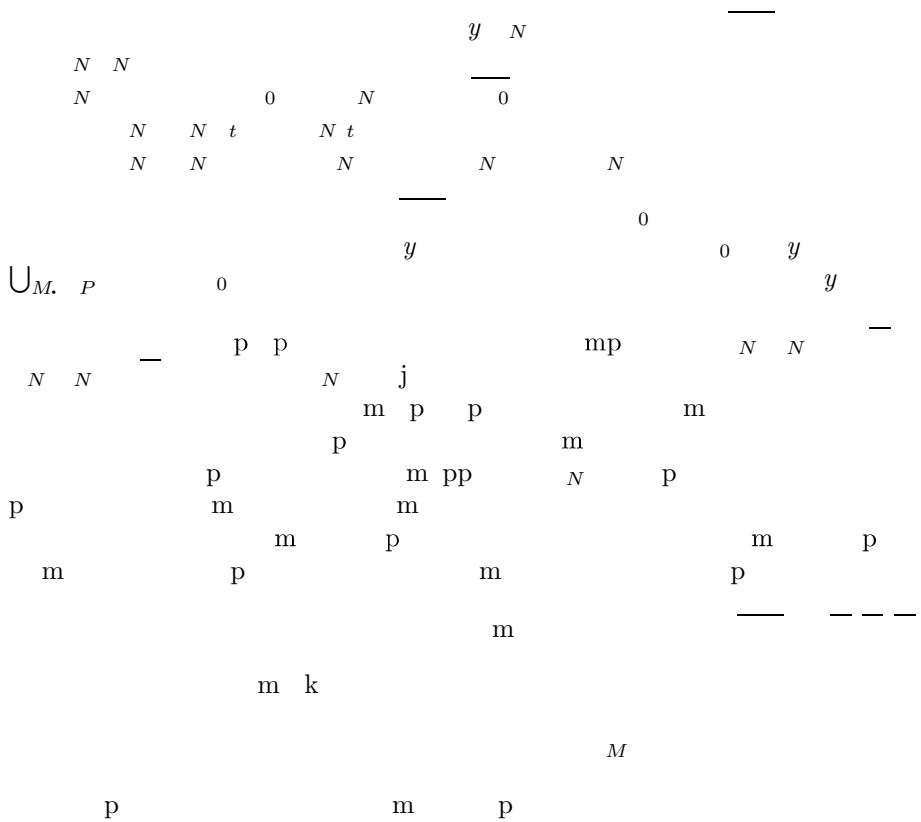
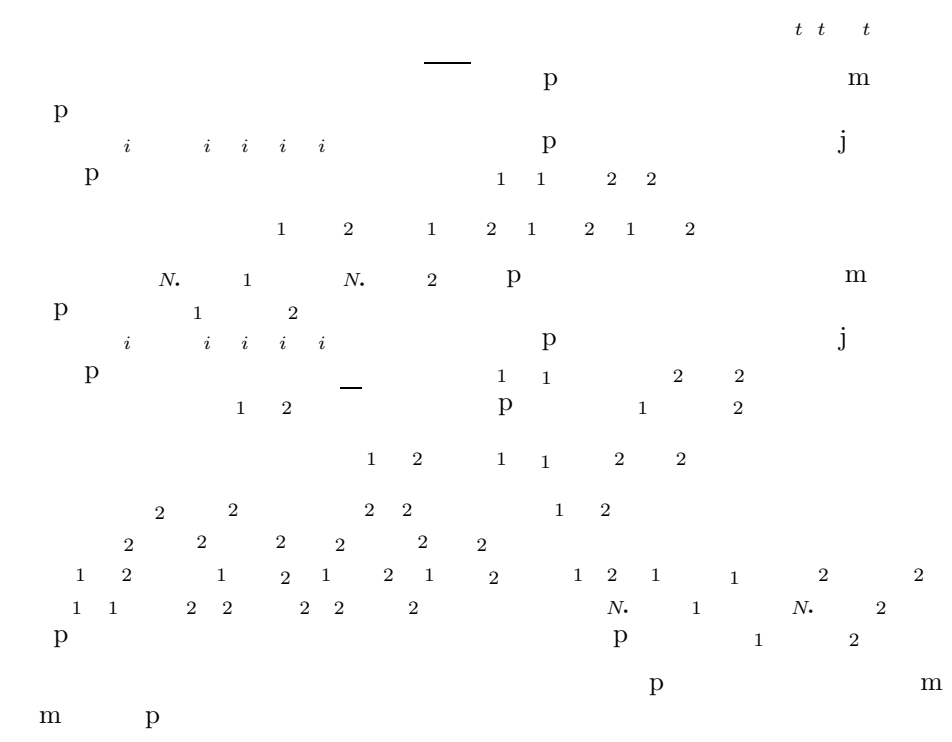
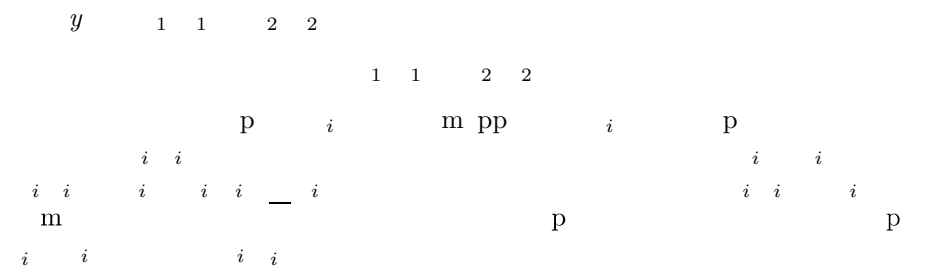


Fig. 4. $\begin{matrix} m & k \\ m & \end{matrix}$ $\begin{matrix} m \\ p \end{matrix}$ $\begin{matrix} m & k \\ m & \end{matrix}$

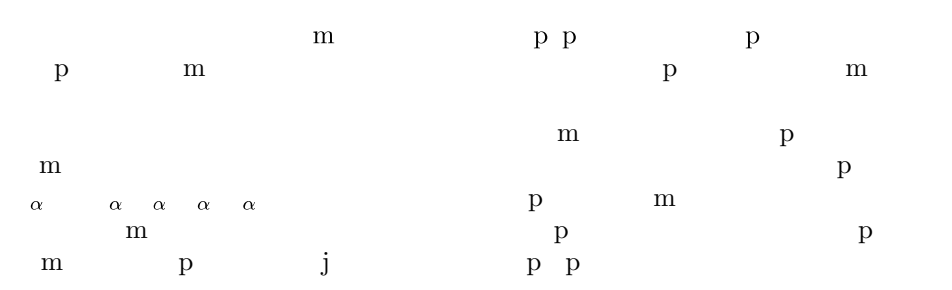




Lemma 3.



5 Relationship between Process Terms and Processes of Elementary Nets



..

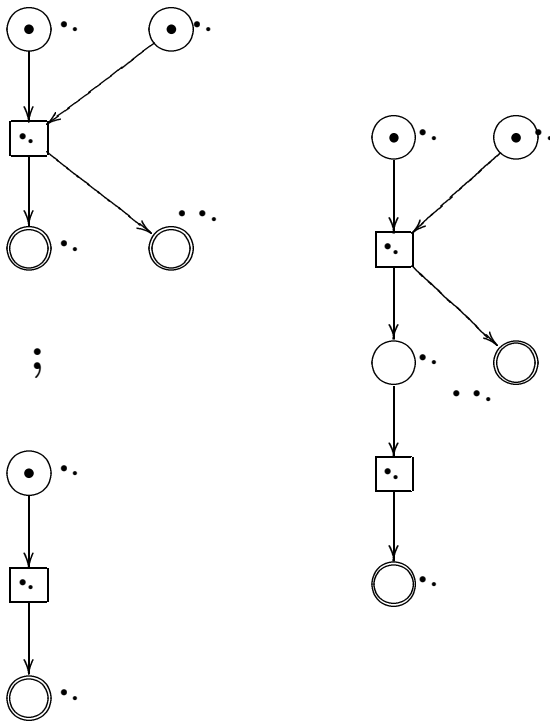


Fig. 5.

p

1

2

m

α α
 α α

α α

fl p m m k

α
m k p

p p

p m

α
p m k
p m
1 2 p 1 2 m m 1 2 p
1 2 1 2 p m
 α α α

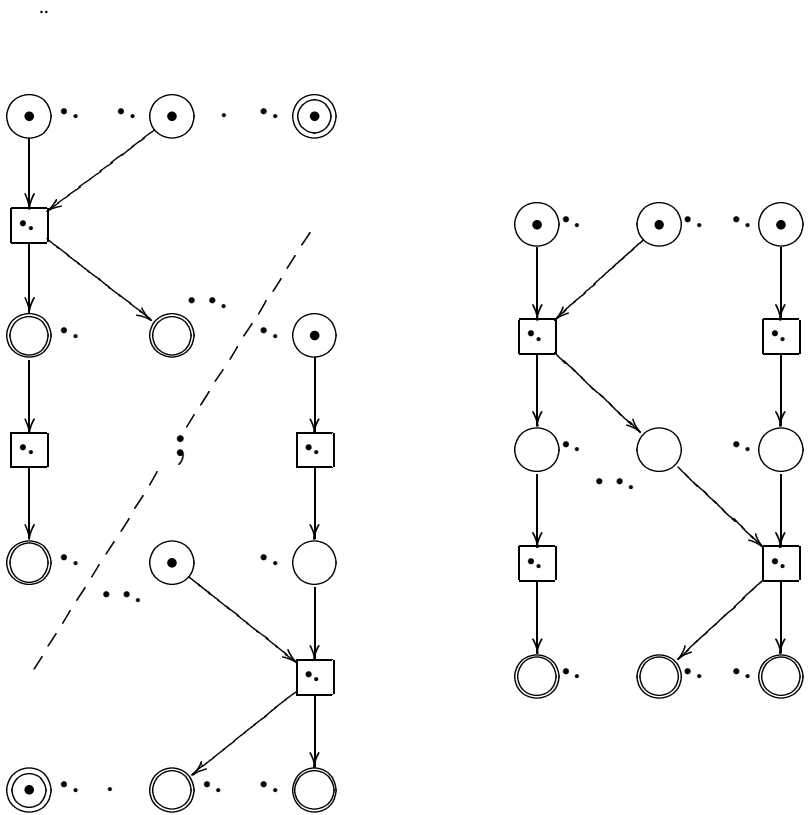


Fig. 7. $\begin{matrix} p & & m & & & & m & m & p \\ 1 & 2 & 5 & 3 & 4 & 1 & & & \end{matrix}$

$\begin{matrix} m & & p & & p & \alpha & m & m & p & p \\ p & & m & & & & & p & & \end{matrix}$

Definition 14. $\begin{matrix} & & p & & & & p \\ p & & m & m & & & p \end{matrix}$ $\begin{matrix} y & \alpha \end{matrix}$

Lemma 4. $\begin{matrix} N & N & N & N & & 1 & 2 & N \\ 1 & 2 & \text{co} & 1 & 2 & & & N & 2 \end{matrix}$

$\begin{matrix} 1 & 2 & \text{co} & \text{mp} & & 1 & 2 & 1 & 2 & & N \\ j & & & & N & 1 & N & 2 & N & 1 & N & 2 & m \end{matrix}$

$\begin{matrix} p & & N & 1 & N & 2 & N & 2 & N & 1 & & & & \end{matrix}$

$\begin{matrix} N & 1 & N & 2 & & & p & 1 & 1 & 2 & 2 \\ N & 1 & N & 2 & & 1 & 2 & \text{co} & 1 & 2 & & & \end{matrix}$

$\begin{matrix} j & & N & & & 2 & 1 & + & 1 & 2 \end{matrix}$

Theorem 1.

Corollary 1.

ary 1. y y

y

m m m p p m

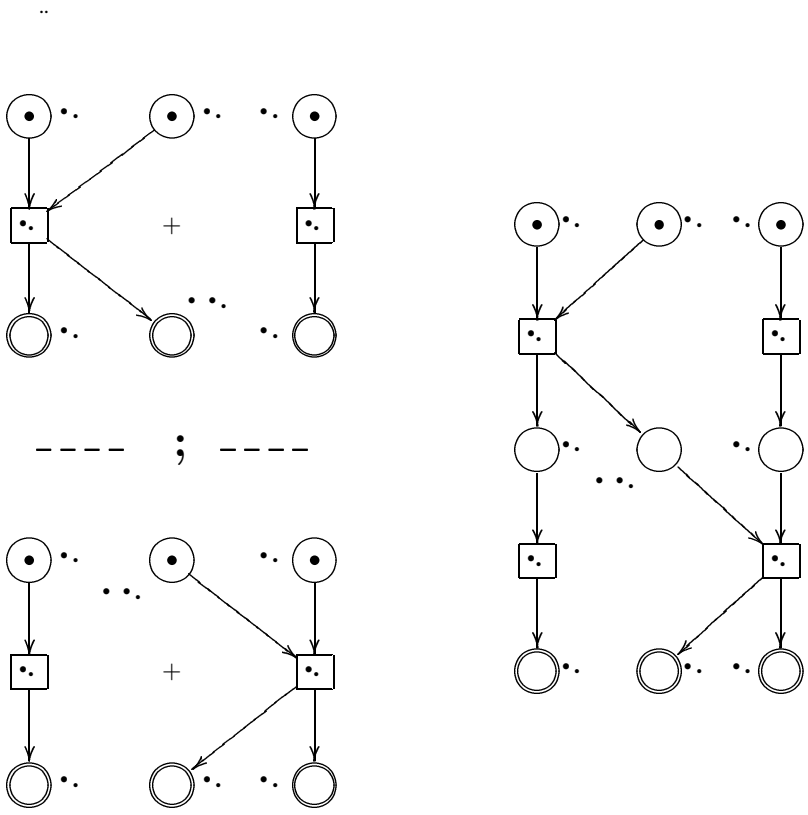
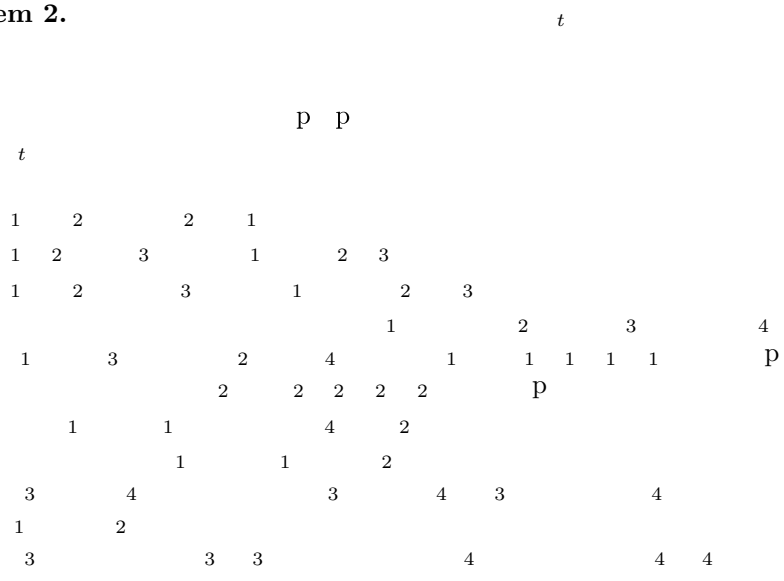


Fig. 8. p m 1 2

3 4 m m p m

Theorem 2.



Theorem 3.

..

Theorem 4. y y

y t y

p m m mp p
 p m p p m p
 p p N mp p
 p mp m p m m k
 j m m p m p p

6 Conclusion

p p p m p p p m
 p p m m p k
 m m p p p

References

..... 9
99
..... 99
..... 99
..... 9 999

The diagram consists of several horizontal rows of points. The points are arranged in a way that suggests a sequence or a progression. Labels '9', '99', and '999' are placed near the rows of points. The labels '9' appear at the bottom left, middle left, and middle right. The label '99' appears in the middle left and middle right. The label '999' appears at the top right. The points are arranged in a way that suggests a sequence or a progression.

User Interface Prototyping Based on UML Scenarios and High-Level Petri Nets¹

Mohammed Elkoutbi and Rudolf K. Keller

Département d'informatique et de recherche opérationnelle
Université de Montréal
C.P. 6128, succursale Centre-ville, Montréal, Québec H3C 3J7, Canada
voice: (514) 343-6782
fax: (514) 343-5834
e-mail: {elkoutbi, keller}@iro.umontreal.ca
<http://www.iro.umontreal.ca/~{elkoutbi, keller}>

Abstract: In this paper, we suggest a requirement engineering process that generates a user interface prototype from scenarios and yields a formal specification of the system in form of a high-level Petri net. Scenarios are acquired in the form of sequence diagrams as defined by the Unified Modeling Language (UML), and are enriched with user interface information. These diagrams are transformed into Petri net specifications and merged to obtain a global Petri net specification capturing the behavior of the entire system. From the global specification, a user interface prototype is generated and embedded in a user interface builder environment for further refinement. Based on end user feedback, the input scenarios and the user interface prototype may be iteratively refined. The result of the overall process is a specification consisting of a global Petri net, together with the generated and refined prototype of the user interface.

Keywords: User interface prototyping, scenario specification, high-level Petri net, Unified Modeling Language.

1 Introduction

Scenarios have been identified as an effective means for understanding requirements [16] and for analyzing human computer interaction [14]. A typical process for requirement engineering based on scenarios [7] has two main tasks. The first task consists of generating from scenarios specifications that describe system behavior. The second task concerns scenario validation with users by simulation and prototyping. These tasks remain tedious activities as long as they are not supported by automated tools.

For the purpose of validation in early development stages, rapid prototyping tools are commonly and widely used. Recently, many advances have been made in user

¹ This work is supported by FCAR (Fonds pour la formation des chercheurs et l'aide à la recherche au Québec) and NSERC (National Sciences and Research Council of Canada).

interface (UI) prototyping tools like UI builders and UI management systems. Yet, the development of UIs is still time-consuming, since every UI object has to be created and laid out explicitly. Also, specifications of dialogue controls must be added by programming (for UI builders) or via a specialized language (for UI management systems).

This paper suggests an approach for requirements engineering that is based on the Unified Modeling Language (UML) and high-level Petri nets. The approach provides an iterative, four-step process with limited manual intervention for deriving a prototype of the UI from scenarios and for generating a formal specification of the system. As a first step in the process, the use case diagram of the system as defined by the UML is elaborated, and for each use case occurring in the diagram, scenarios are acquired in the form of UML sequence diagrams and enriched with UI information. In the second step, the use case diagram and all sequence diagrams are transformed into Colored Petri Nets (CPNs). In step three, the CPNs describing one particular use case are integrated into one single CPN, and the CPNs obtained in this way are linked with the CPN derived from the use case diagram to form a global CPN capturing the behavior of the entire system. Finally, in step four, a prototype of the UI of the system is generated from the global CPN and embedded in a UI builder environment for further refinement.

In our previous work, we have investigated and implemented the generation of UI prototypes from UML scenarios using exclusively the UML, most notably UML Statecharts [5, 13, 21, 22]. In this Statechart-based approach, Statecharts are used to integrate the UML scenarios and capture object and UI behavior. In the work presented in this paper, we decided to take a Petri-net based approach, with CPNs taking the role of UML Statecharts. We opted for Petri nets because of their strong support of concurrency, their ability to capture and simulate multiple copies of scenarios in the same specification, and for the wealth of available tools for analyzing, simulating, and verifying Petri nets. A comparison of the two approaches is provided in Section 5 of the paper.

In our approach, we aim to model separately the use case and the scenario levels. We also want to keep track of scenarios after their integration. Thus, we need a PN class that supports hierarchies as well as colors or objects to distinguish between scenarios in the resulting specification. We adopted Jensen's definition of CPN [10] which is widely accepted and supported by the *designCPN* tool [3] for editing, simulating, and verifying CPNs. Object PNs could also have been used, but CPNs are largely sufficient for this work. In our current implementation, UI prototyping is embedded into the *Visual Café* environment [23] for further refinement.

Section 2 of this paper gives a brief overview of the UML diagrams relevant to our work and introduces a running example. In Section 3, the four activities leading from scenarios to executable UI prototypes are detailed. Section 4 reviews related work, and in Section 5, we discuss a number of issues related to the proposed approach. Section 6 concludes the paper and provides an outlook into future work.

2 Unified Modeling Language

The UML [19], which is emerging as a standard language for object-oriented modeling, provides a syntactic notation to describe all major views of a system using

different kinds of diagrams. In this section, we discuss the three UML diagrams that are relevant for our work: Class diagram (ClassD), Use Case diagram (UsecaseD), and Sequence diagram (SequenceD). As a running example, we have chosen to study a part of an extended version of the Automatic Teller Machine (ATM) system described in [2].

2.1 Class Diagram (ClassD)

The ClassD represents the static structure of the system. It identifies all the classes for a proposed system and specifies for each class its attributes, operations, and relationships to other objects. Relationships include inheritance, association, and aggregation. The ClassD is the central diagram of UML modeling. Figure 1 depicts the ClassD for the ATM system.

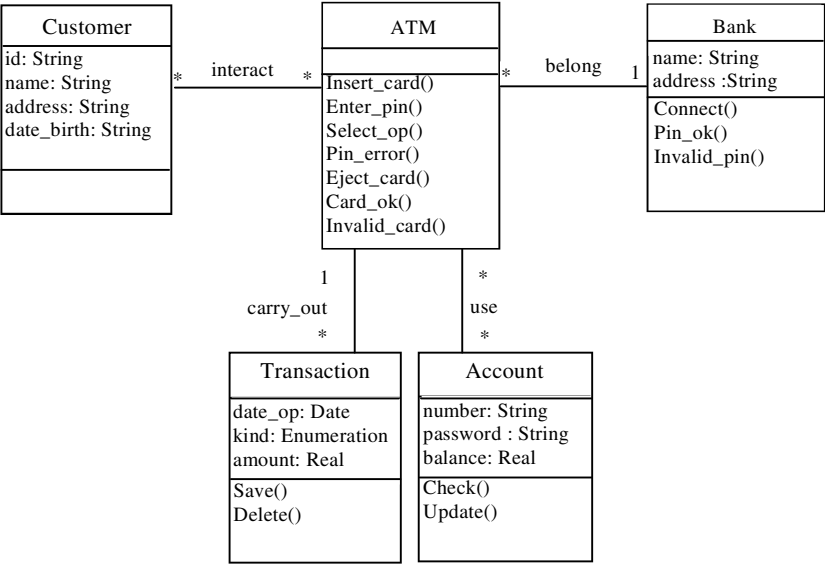


Figure 1: Class diagram of the ATM system.

2.2 Use Case Diagram (UsecaseD)

The UsecaseD is concerned with the interaction between the system and actors (objects outside the system that interact directly with it). It presents a collection of use cases and their corresponding external actors. A use case is a generic description of an entire transaction involving several objects of the system. Use cases are represented as ellipses, and actors are depicted as icons connected with solid lines to the use cases they interact with. One use case can call upon the services of another use case. Such a relation is called a *uses* relation and is represented by a directed solid line. Figure 2 shows as an example the UsecaseD corresponding to the ATM system. The *extends*

relation, which is also defined in UsecaseDs, can be seen as a *uses* relation with an additional condition upon the call. A UsecaseD is helpful in visualizing the context of a system and the boundaries of the system's behavior. A given use case is typically characterized by multiple scenarios.

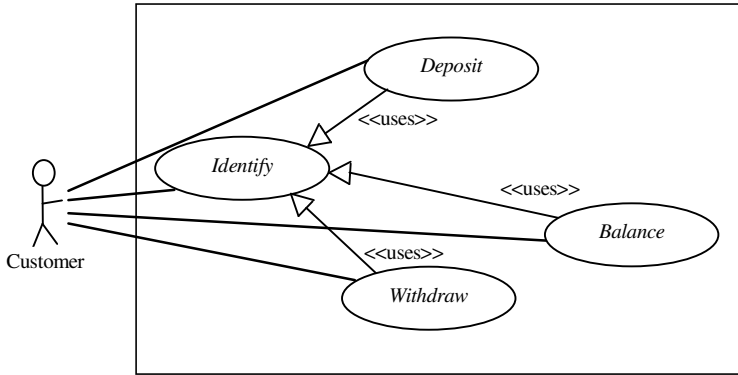


Figure 2: Use Case diagram of the ATM system.

2.3 Sequence Diagram (SequenceD)

A scenario shows a particular series of interactions among objects in a single execution (*instance*) of a use case. Scenarios can be viewed in two different ways: through SequenceDs or collaboration diagrams. Both types of diagrams rely on the same underlying semantics, and conversion from one to the other is possible. For our work, we chose to use SequenceDs for their simplicity and wide use.

A SequenceD shows the interactions among the objects participating in a scenario in temporal order. It depicts the objects by their lifelines and shows the messages they exchange in time sequence. However, it does not capture the associations among the objects. A SequenceD has two dimensions: the vertical dimension represents time, and the horizontal dimension represents the objects. Messages are shown as horizontal solid arrows from the lifeline of the object sender to the lifeline of the object receiver. A message may be guarded by a condition, annotated by iteration or concurrency information, and/or constrained by an expression. Constraints are used in our work to enrich messages with UI information.

Figures 3 and 4 depict two SequenceDs of the use case *Identify*. Figure 3 represents the scenario where the customer is correctly identified (*regularIdentify*), whereas Figure 4 shows the case where the customer entered an incorrect pin (*errorIdentify*).

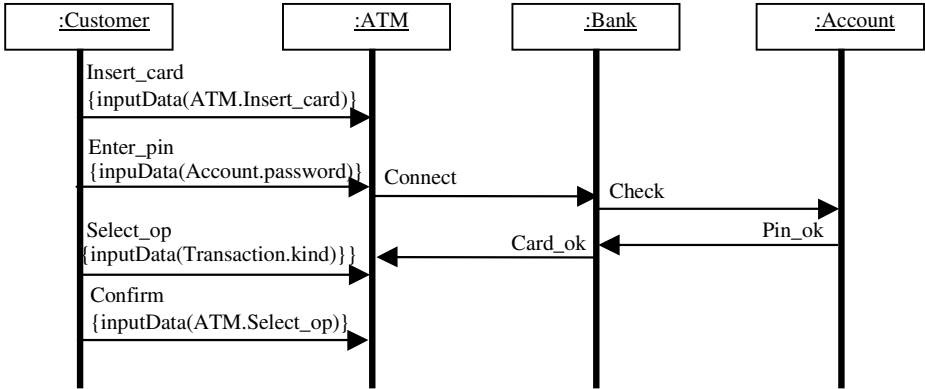


Figure 3: Scenario *regularIdentify* of the use case *Identify*.

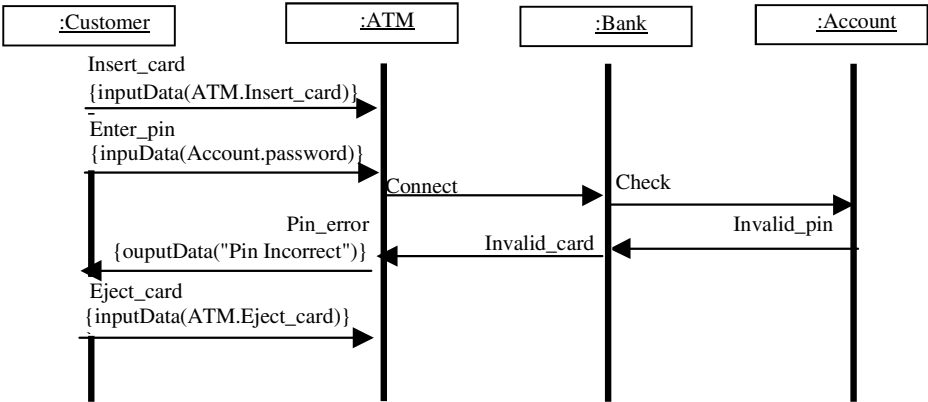


Figure 4: Scenario *errorIdentify* of the use case *Identify*.

Beyond the UML standard message constraints found in SequenceDs, we define the two additional constraints *inputData* and *outputData*. The *inputData* constraint indicates that the corresponding message holds input information from the user. The *outputData* constraint specifies that the corresponding message carries information for display. Both *inputData* and *outputData* constraints have a parameter that indicates the kind of user action. This parameter normally represents the dependency between the message and the elements of the underlying ClassD. It may be either a method name, one or several class attributes, or a string literal (see figures 3 and 4).

Once the analyst has specified the UI constraints of the messages in the SequenceD at hand, this information is used to determine the corresponding widgets that will appear in the UI prototype. Widget generation adheres to a list of rules, which is based on the terminology, heuristics and recommendations found in [8] and which includes the following eight items:

- A button widget is generated for an *inputData* constraint with a method as dependency, e.g., *Insert_card()* {*inputData*(*ATM.insert_card()*)} in Figure 3.
- An enabled textfield widget is generated in case of an *inputData* constraint with a dependency to an attribute of type String, Real, or Integer, e.g., *Enter_pin()* {*inputData*(*Account.password()*)} in Figure 3.
- A group of radio buttons widgets are generated in case of an *inputData* constraint with a dependency to an attribute of type Enumeration having a size less than or equal to 6, e.g., *Select_op()* {*inputData*(*Transaction.kind()*)} in Figure 3.
- An enabled list widget is generated in case of an *inputData* constraint with a dependent attribute of type Enumeration having a size greater than 6 or with a dependent attribute of type collection.
- An enabled table widget is generated in case of an *inputData* constraint with multiple dependent attributes.
- A disabled textfield widget is generated for an *outputData* constraint with a dependency to an attribute of type String, Real, or Integer.
- A label widget is generated for an *outputData* constraint with no dependent attribute, e.g., *Pin_error()* {*outputData*("Pin Incorrect")} in Figure 4.
- A disabled list widget is generated in case of an *outputData* constraint with a dependent attribute of type Enumeration having a size greater than 6 or with a dependent attribute of type collection.
- A disabled table widget is generated in case of an *outputData* constraint with multiple dependent attributes.

3 Description of Approach

In this section, we detail the iterative process for deriving a system UI prototype from scenarios using the UML and CPNs. Figure 5 presents the sequence of activities involved in the proposed process.

In the *Scenario Acquisition* activity, the analyst elaborates the UsecaseD, and for each use case, he or she elaborates several SequenceDs corresponding to the scenarios of the use case at hand. The *Specification Building* activity consists of deriving CPNs from the acquired UsecaseD and SequenceDs. During *Scenario Integration*, CPNs corresponding to the same use case are iteratively merged to obtain an integrated CPN of the use case. Integrated CPNs serve as input to both the *CPN Verification* and the *UI Prototype Generation* activities. During *Prototype Evaluation*, the generated prototype is executed and evaluated by the end user. In the *CPN Verification* activity, existing algorithms can be used to check behavioral properties.

In the following subsections, we will discuss in detail the four activities of the UI prototyping process: scenario acquisition, specification building, scenario integration, and UI prototype generation. The CPN verification activity is discussed in [4].

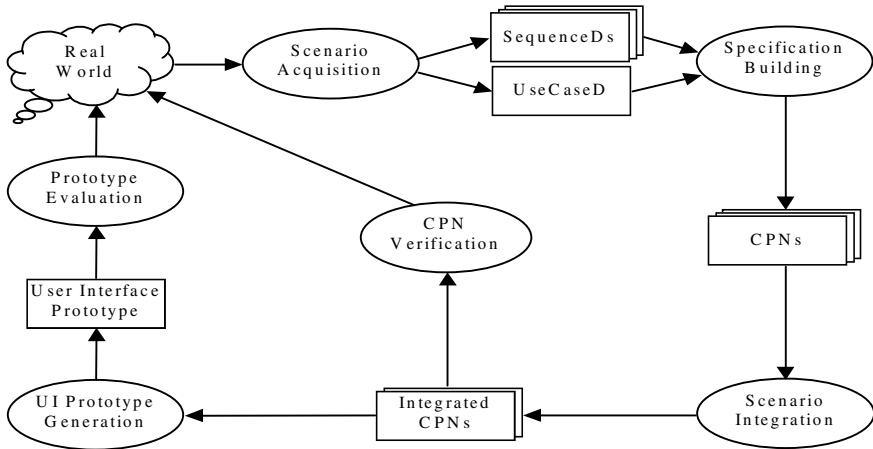


Figure 5: Activities of the proposed process.

3.1 Scenario Acquisition

In this activity, the analyst elaborates the UsecaseD capturing the system functionalities, and for each use case, he or she acquires the corresponding scenarios in form of SequenceDs. For instance, the UsecaseD of the ATM system is shown in Figure 1, and two SequenceDs of the use case *Identify* are given in figures 3 and 4.

Scenarios of a given use case are classified by type and ordered by frequency of use. We have considered two types of scenarios: normal scenarios, which are executed in normal situations, and scenarios of exception executed in case of errors and abnormal situations. The frequency of use (or the frequency of execution) of a scenario is a number between 1 and 10 assigned by the analyst to indicate how often a given scenario is likely to occur. In our example, the use case *Identify* has one normal scenario (scenario *regularIdentify* with frequency 10) and a scenario of exception (scenario *errorIdentify* with frequency 5). This classification and ordering is used for the composition of UI blocks [5].

3.2 Specification Building

This activity consists of deriving CPNs from both the acquired UsecaseD and all the SequenceDs. These derivations are explained below in the subsections *Use case specification* and *scenario specification*.

3.2.1 Use Case Specification

The CPN corresponding to the UsecaseD is derived by mapping use cases into places. The transition leading to one place (*Enter*) corresponds to the initiating action of the use case. A place *Begin* is always added to model the initial state of the system. After a use case execution, the system will return, via an *Exit* transition, back to its initial state for further use case executions. The place *Begin* may contain several tokens to

model concurrent executions. Figure 6 depicts the CPN derived from the ATM system's UsecaseD (Figure 2).

In a UsecaseD, a use case can call upon the services of another one via the relation *uses*. This relation may have several meanings depending on the system being modeled. Consider a use case Uc_i using a use case Uc_j . Figure 7(a) shows the general form of this relation. The use case Uc_i is decomposed into three sub-use cases: Uc_{i1} represents the part of Uc_i executed before the call of Uc_j , Uc_{i2} represents the part of Uc_i that is concurrently executed with Uc_j , and Uc_{i3} represents the part executed after the termination of Uc_j . Note that one or two of these three sub-use cases may be empty. The figures 7(a) through 7(g) depict the eight possible mappings.

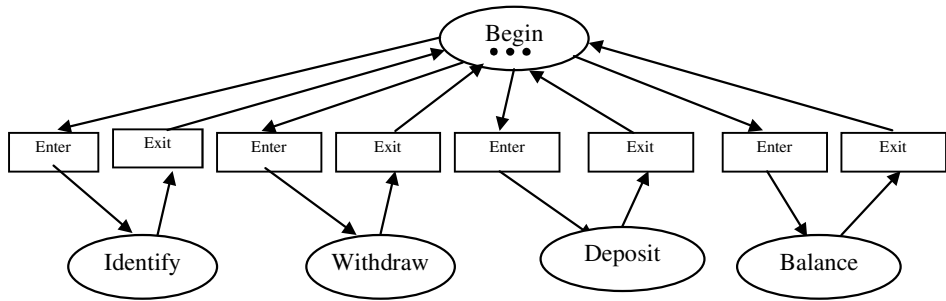


Figure 6: CPN derived from the UsecaseD of the ATM system.

A relation of type (g) between Uc_i and Uc_j means that Uc_j precedes Uc_i . This implies that Uc_i is not directly accessible from the place *Begin*. So the transitions from the place *Begin* to the place representing Uc_i must be substituted for an *Enter* transition into Uc_j and an *Enter* transition from Uc_i into Uc_j . In the ATM system (Figure 1), all three *uses* relations are of type (g), and the initial CPN (Figure 7) must be updated accordingly (Figure 9(a)).

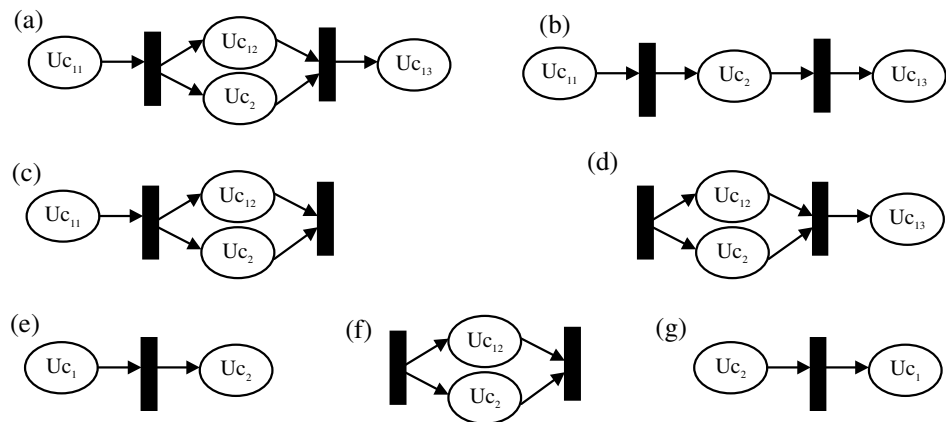


Figure 7: Possible mappings of the *uses* relation (Uc_i, Uc_j).

The designCPN tool, which we adopted in our work, allows for the refinement of transitions, but does not support the refinement of places. Therefore, in order to substitute the use cases, which are represented as places, for CPNs representing integrated scenarios (see Subsection 3.3), the CPN obtained after processing the *uses* relation (Figure 8(a)) requires adaptation: each subnet $Enter \rightarrow place_i \rightarrow Exit$ is substituted for a simple transition representing the use case underlying $place_i$ (cf. dashed ellipse in Figure 8(a)), and intermediary places such as *endIdentify* are inserted (see Figure 8(b)).

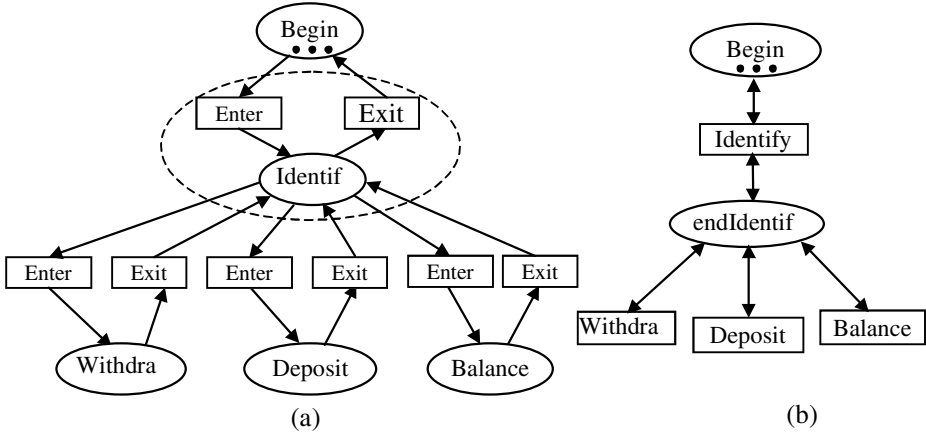


Figure 8: CPN of the UsecaseD of the ATM system after (a) processing the *uses* relation, and (b) adaptation to designCPN.

3.2.2 Scenario Specification

For each scenario of a given use case, the analyst builds an associated table of object states. This table is directly obtained from the SequenceD of the scenario by following the exchange of messages from top to bottom and identifying the changes in object states caused by the messages. For example, tables 1 and 2 show the object state tables associated with the scenarios *regularIdentify* (Figure 3) and *errorIdentify* (Figure 4). In such tables, a scenario state is represented by the state vector of the objects participating in the scenario (column *Scenario state* in Table 1 and 2, resp.).

From each object state table, a CPN is generated by transforming scenario states into places, and messages into transitions (see figures 9 and 10)². Each scenario is assigned a distinct color, e.g., *rid* for the *regularIdentify* scenario, and *eid* for the *errorIdentify* scenario. All CPNs (scenarios) of the same use case will have the same initial place (state) which we call *B* in figures 9 and 10. This place will serve to link the integrated CPN (see below) with the CPN modeling the UsecaseD of the system (Figure 8(b)).

² For readability, we do not show screen dumps produced by designCPN, but replace them with CPNs redrawn by hand.

Objects Messages	Cus- tomer	ATM	Bank	Account	Scenario state
Insert_card	Present	Card_in	void	void	S1={Present, Card_in, void, void}
Enter_pin	Present	Pin-entered	void	void	S2={Present, Pin_entered, void, void}
Connect	Present	Pin-entered	Connected	void	S3={Present, Pin_entered, Connected, void}
Check	Present	Pin-entered	Connected	Checked	S4={Present, Pin_entered, Connected, Checked}
Pin_ok	Present	Pin-entered	Valid_pin	Checked	S5={Present, Pin_entered, Valid_pin, Checked}
Card_ok	Present	Valid-card	Valid_pin	Checked	S6={Present, Valid_card, Valid_pin, Checked}
Select_op	Present	Selection	Valid_pin	Checked	S7={Present, Selection, Valid_pin, Checked}
Confirm	Present	Confir- mation	Valid_pin	Checked	S8={Present, Confirmation, Valid_pin, Checked}

Table 1: Object state table associated with the scenario *regularIdentify*.

Objects Messages	Cus- tomer	ATM	Bank	Account	Scenario state
Insert_card	Present	Card_in	void	void	S1={Present, Card_in, void, void}
Enter_pin	Present	Pin-entered	void	void	S2={Present, Pin_entered, void, void}
Connect	Present	Pin-entered	Connected	void	S3={Present, Pin_entered, Connected}
Check	Present	Pin-entered	Connected	Checked	S4={Present, Pin_entered, Connected, Checked}
Invalid_pin	Present	Pin-entered	Invalid_pin	Checked	S9={Present, Pin_entered, Invalid_pin, Checked}
Invalid_card	Present	Invalid-card	Invalid_pin	Checked	S10={Present, Invalid_card, Invalid_pin, Checked}
Pin_error	Present	Invalid_pin	Invalid_pin	Checked	S11={Present, Invalid_pin, Invalid_pin, Checked}
Eject_card	Present	Idle	Invalid_pin	Checked	S12={Present, Idle, Invalid_pin, Checked}

Table 2: Object state table associated with the scenario *errorIdentify*.

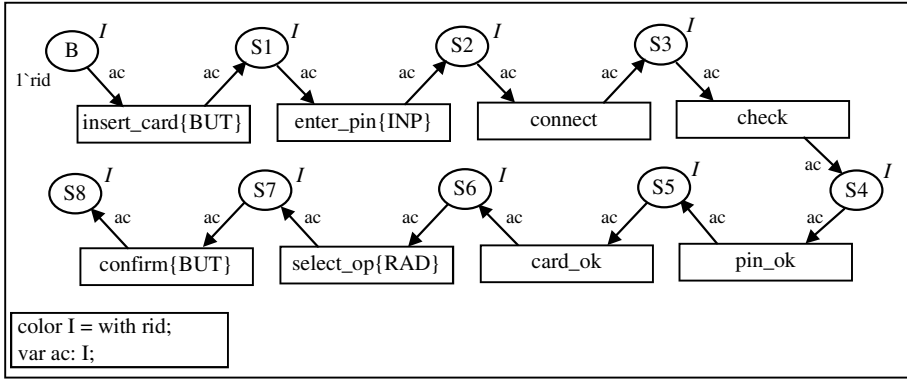


Figure 9: CPN corresponding to the *regularIdentify* scenario.

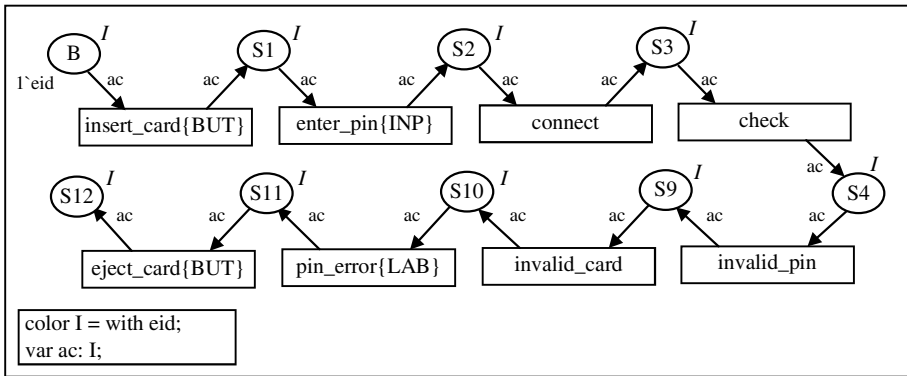


Figure 10: CPN corresponding to the *errorIdentify* scenario.

Note that in scenario specification, only the building of the object state tables is manual. The rest of the operation is fully automatic.

The next activity of the approach, scenario integration, requires as input serialized CPNs. Since designCPN uses SGML as interchange format and since conversion tools between SGML, XML, and Java are readily available, we decided to represent CPNs in XML. In our current implementation, we use the transformation scheme as depicted in Figure 11. Using the designCPN tool, the analyst will edit and then save the use case CPN and all scenario CPNs into the textual SGML format. Then, the *SX* tool [1] is used to do the conversion from SGML to rough XML (RXML), and the *XJParse* Java program [9] to convert RXML into XML.

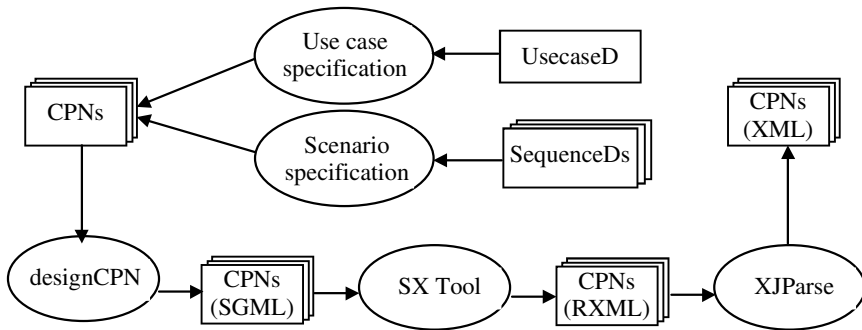


Figure 11: Generation of CPNs in XML format from UsecaseD and SequenceDs.

3.3 Scenario Integration

In this activity, we aim to merge all CPNs corresponding to the scenarios of a use case Uc_i , in order to produce an integrated CPN modeling the behavior of the use case. Our algorithm is based on a preliminary version presented in [6]. It takes an incremental approach to integration. Given two scenarios with corresponding CPNs CPN_1 and CPN_2 , the algorithm merges all places in CPN_1 and CPN_2 having the same names. The merged places will have as color the union of the colors of the two scenarios. Then, the algorithm looks for transitions having the same input and output places in the two scenarios and merges them with an *OR* between their guard conditions. In the following, we describe the algorithm in pseudocode, using the “dot”-notation known from object-oriented languages.

```

Uci.IntergrateScenarios()
  scList = Uci.getListOfScenarios();
  // returns the list of scenarios of the use case Uci
  uc_cpn = getXML(scList[0]);
  // returns the XML file corresponding to the scenario scList[0].
  i=1;
  while (i < scList.size())
    sc_cpn = getXML(scList(i));
    sc_cpn = makeUniqueID( uc_cpn, sc_cpn);
    uc_cpn = merge(uc_cpn,sc_cpn);
    i = i + 1;
  end
end Uci.IntergrateScenarios

```

XML identifies each element in a CPN (place, transition, edge, etc.) by a distinct identifier (ID). Before integrating two scenarios, the method `makeUniqueID` checks if both the two input files `uc_cpn` and `sc_cpn` comprise distinct IDs. In case they share common IDs, `makeUniqueID` will modify the IDs of `sc_cpn` by adding the maximum ID of `uc_cpn` to all IDs of `sc_cpn`.

Merging (integrating) two scenarios whose CPNs have the colors $[sc_1]$ and $[sc_2]$, respectively, will produce a CPN with the list $[sc_1, sc_2]$ as color. The operation of merging follows the steps described below:

```

merge(uc_cpn, sc_cpn)
  uc_cpn.addPlaces(sc_cpn)
  // adds in uc_cpn places of sc_cpn that do not exist in uc_cpn
  for each t in sc_cpn.getListOfTransitions()
    t' = uc_cpn.LookForTrans(t)
    // t' is a transition of uc_cpn with  $\bullet t = \bullet t'$  and  $t \bullet = t' \bullet$ 
    if (t' does not exist)
      uc_cpn.addtrans(t)
    endif
  end
end
uc_cpn.addEdges(sc_cpn)
// adds to uc_cpn edges of sc_cpn that do not exist in uc_cpn
uc_cpn.mergeColors(sc_cpn)
// calculates the new color of the integrated CPN (uc_cpn)
uc_cpn.putColorsOnPlaces(sc_cpn)
// all places of the net will have the merged color
uc_cpn.putGuardOnTransitions(sc_cpn)
// common transitions will be guarded by the merged color,
// the others will be guarded by their original colors
uc_cpn.putVariablesOnEdges(sc_cpn)
// put on edges variables or token expressions
end merge

```

The result of applying this algorithm on the two scenarios of the use case *Identify* (figures 9 and 10) is shown in Figure 12.

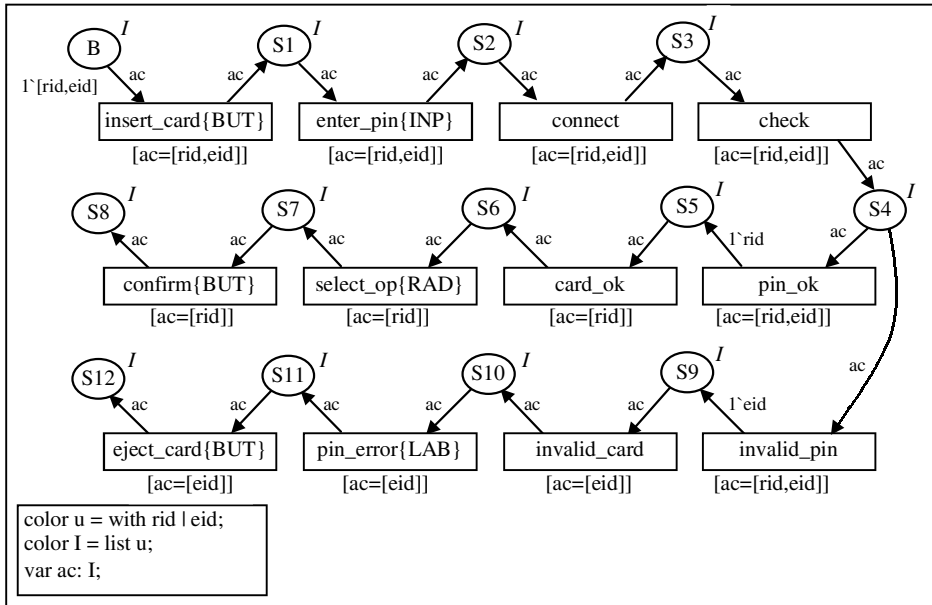


Figure 12: CPN of the use case *Identify* after merging the scenarios *regularIdentify* and *errorIdentify*.

When integrating several scenarios, the resulting specification captures in general not only the input scenarios, but perhaps even more. Figure 13 gives an example illustrating this issue: the resulting scenario Sc will capture the initial scenarios Sc_1 (T_1, T_2, T_3, T_4, T_5) and Sc_2 (T_1, T_6, T_7, T_4, T_9), as well as the two new scenarios

$(T_1, T_2, T_3, T_4, T_9)$ and $(T_1, T_6, T_7, T_4, T_5)$. After integrating the two scenarios, the initial place B (see Figure 13) will be shared, yet we do not know which scenario will be executed, and neither the color of Sc_1 nor the color of Sc_2 can be assigned to B . This problem was described by Koskimies and Makinen [12], and we refer to it as *interleaving problem* [6].

To solve the interleaving problem, we introduce a *chameleon token*, i.e., a token that can take on several colors [6]. Using designCPN, a chameleon token is modeled by a list of colors. Upon visiting the places of the integrated net, it will be marked by the intersection of its colors and the colors of the place being visited. When the token passes to the place S_1 , it keeps the composite color $[sc_1, sc_2]$, and if it passes from S_1 to S_2 , its color changes to $[sc_1]$ and will remain unchanged for the rest of its journey, or if it passes from S_1 to S_6 , its color changes to $[sc_2]$ and will remain unchanged for the rest of its journey.

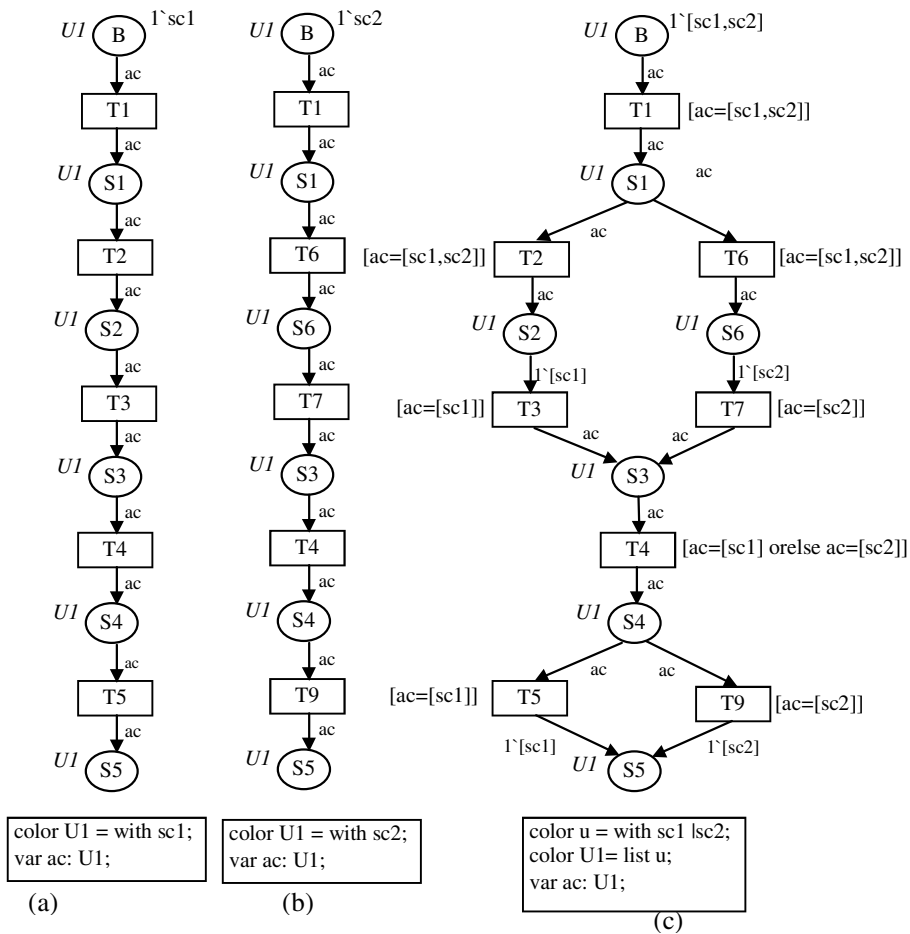


Figure 13: Interleaving problem of scenario integration: (a) scenario Sc_1 , (b) scenario Sc_2 and (c) integrated scenario Sc .

Transitions that belong to only one of the scenarios Sc_1 and Sc_2 will be guarded by the color of their respective scenarios (see T_3, T_5, T_7 , and T_9 in Figure 13(c)). But this is not the case for the transitions T_2 and T_6 which are required to transform tokens from the composite color (list of colors) to a single color. Therefore, they must be guarded by the list of colors. For transitions that are shared by the two scenarios Sc_1 and Sc_2 , they will be guarded by the composite color (see T_1 in Figure 13(c)) or by a disjunction of single colors of scenarios (see T_4 in Figure 13(c)).

An integrated CPN corresponding to a given use case can be connected to the CPN derived from the UsecaseD through a transition that is appended to the place B of the integrated CPN. This transition transforms the uncolored tokens of the CPN of the UsecaseD to the composite color of the integrated CPN.

3.4 User Interface Prototype Generation

In this activity, we derive from the CPN specifications a UI prototype of the system. The generated prototype is standalone and comprises a menu to switch between the different use cases. The various screens of the prototype represent the static aspect of the UI; the dynamic aspect of the UI, as captured in the CPN specifications, maps into the dialog control of the prototype. In our current implementation, prototypes are Java applications comprising each a number of frames and navigation functionality (see Figure 14).

The activity of prototype generation is composed of the following five operations which are described in detail in [6].

- Generating graph of transitions.
- Masking non-interactive transitions.
- Identifying UI blocks.
- Composing UI blocks.
- Generating frames from composed UI blocks.

These operations follow closely the corresponding operations in the Statechart-based approach [6], except for the operation of generating the graph of transitions. This operation consists of deriving a directed graph of transitions (GT) from the CPN of a given use case. Transitions of the CPN will represent the nodes of the GT, and edges will indicate the precedence of execution among transitions: if two transitions T_1 and T_2 are consecutively executed, there will be an edge between the nodes representing T_1 and T_2 .

A GT has a list of nodes *nodeList*, a list of edges *edgeList*, and a list of initial nodes *initialNodeList* (entry nodes of the graph). The list of nodes *nodeList* of a GT is easily obtained since it corresponds to the transition list of the CPN at hand. The list of edges *edgeList* of a GT is obtained by linking the transitions in $\bullet p$ with the ones in $p \bullet$ for each place p in the CPN.

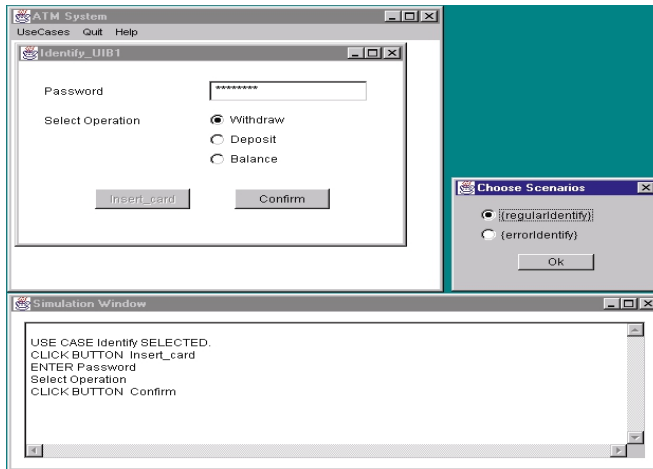


Figure 14: Prototype execution of the ATM system.

To support prototype execution, a *Simulation Window* is generated (Figure 14, bottom window), as well as a dialog box to *Choose Scenarios* (Figure 14, middle-right window). For example, after selecting the use case *Identify* from the *UseCases* menu (Figure 14, top window), a message is displayed in the simulation window that confirms the use case selection and prompts the user to click the button *Insert_card*. When this button is clicked, the fields *Password* and *Select Operation* are enabled. Then, the simulator prompts the user for information entry. When execution reaches a node in GT from which several continuation paths are possible (e.g., button *Confirm* clicked), the prototype displays the dialog box for scenario selection. In the example of Figure 14, the upper selection corresponds to the scenario *regularIdentify* and the lower one to the scenario *errorIdentify*. Once a path has been selected, the traversal of GT continues.

4 Related Work

In this section, we review some related work in the area of UI specification and automatic generation based on Petri nets. Each of the three approaches presented below suggests some high-level UI specification, yet none of them generates these specifications from scenario descriptions. For a discussion of work dealing with the simulation of specifications and with UI generation from specifications other than Petri nets refer to [5].

In the *TADEUS* approach [20] (TAsk-based DEvelopment of User interface Software), the development of the UI follows three stages: the requirements analysis and specification, the dialogue design, and the generation of the UI prototype. During the dialogue design stage, two levels of dialogues are manually built: the navigation dialogue that describes the sequencing between different task presentation units (dialogue views), and the processing dialogue that describes the changes on UI objects inside a dialogue view. The result of generation is a script file for an existing UI management system. Dialogue graphs is the formalism used to specify dialogues.

It is Petri net based, and it permits to model easily different types of dialogues (single, multi, modal, etc.). In this approach, dialogues are manually defined by the UI developer.

Palanque et al. [15] provide the Interactive Cooperative Objects (ICO) formalism for designing interactive systems. Using ICO, an object of the system has four components: a data structure, a set of operations representing the services offered by the object, an object control structure defining the object behavior with high-level Petri nets, and a presentation structured as a set of widgets. The windows of the UI and their interrelations are modeled with ICO objects. This approach is completely manual, whereas in our approach the only manual operation is the building of the object state tables (see Section 3.2.2).

De Rosi et al. [17] propose a task-based approach using Petri nets to describe interactive behavior. They represent a UI by a CPN where transitions are the tasks of the system and places represent the information display after a task execution. A logical projection function is associated to places and transitions of the CPN in order to describe in form of conditions the user actions and the information displayed as a result of performing actions. To complete the UI description, the designer links the physical aspect of the interface to the CPN by means of a physical projection function. This work focuses on specifying the UI, yet does not address UI generation.

5 Discussion of Approach

Below, we discuss our approach in respect to the following points: scenario-based approach, system and object views, visual formalisms for modeling interactive systems, and validation of approach.

5.1 Scenario-Based Approach

Our approach to UI generation exhibits the advantages of scenario-based techniques. In contrast to many data-oriented methods, UIs are generated from specifications describing dynamic system behavior, which are derived from task analysis. Once they are generated, data specifications may be used as the basis for further refinements. In line with Rosson [18] who advocates a “model-first, middle-out design approach” that interleaves the modeling of objects and scenarios, we put the emphasis on the (dynamic) modeling aspect, and generate the dynamic core of UIs rather than focus on screen design and the user-system dialog.

As scenarios are partial descriptions, there is a need to elicit all possible scenarios of the system to produce a complete specification. In our approach, colors of Petri nets are used to inhibit scenario interleaving, that is, the resulting specifications will capture exactly the input scenarios. The integration algorithm can be configured to allow scenario interleaving and to capture more than the mere input scenarios. In this way, new scenarios may be generated from already existing ones.

It is well known that scalability is an inherent problem when dealing with scenarios for large applications. Our approach eases this problem by integrating scenarios on a per use case basis, rather than treating them as an unstructured mass of scenarios.

5.2 System and Object Views

In this paper, we focus on using scenarios for the restructuring of the formal system specification (*system view*). In our previous work [5, 13], we have addressed the specification of individual objects in interactive systems (*object view*). In a scenario-based approach, the specification of the behavior of a given object can be seen as the projection of the set of acquired scenarios on that object. When projecting scenarios onto an object, only messages entering and exiting the object are considered. In this way, the sequence order of messages in the scenarios is lost, and this can lead to capture undesirable behaviors. Furthermore, in an object view, all interface objects must be explicitly identified in the underlying set of scenarios. On the other hand, in a system view, there is no loss of precision, and the UI can be generated from the system specification without the need to explicitly identify UI objects.

For the purpose of UI generation, a system view is appropriate when in all use cases and associated scenarios only one user interacts with the system. In the case of collaborative tasks (more than one user interacts with the use cases), however, an object view will be more suitable.

5.3 Visual Formalisms for Modeling Interactive Systems

Petri nets and Statecharts are among the most powerful visual formalisms used for specifying complex and interactive systems. Statecharts are an extension of state machines to include hierarchies by allowing state refinement, and concurrency by describing independent parts of a system. Concurrency in Statecharts is modeled via orthogonal states. An orthogonal state is a Cartesian product of two or more Statecharts (threads). Sending messages between threads of an orthogonal state is forbidden. Leaving a state of a thread of an orthogonal state leads to exit all threads of this orthogonal state. These restrictions do not apply to the Petri net formalism. Petri nets are known for their support of pure concurrency, all transitions having a sufficient number of tokens in their input places may concurrently be fired. Petri nets in their basic form do not support hierarchy, but the extension of Petri nets used in tools such as designCPN allow for hierarchies in the specification.

Non-deterministic choices can more easily be modeled using Petri nets than based on Statecharts. When two or more transitions share the same input places, the system chooses randomly to fire one of these transitions. In Statecharts, non-deterministic choices cannot directly be modeled because messages are ordered and broadcasted to all concurrent threads.

In many UIs, we have to model exclusive executions (modal windows). This can easily be done using the history states of Statecharts. When entering a modal window, an event must be broadcasted to all concurrent threads to enter their history states. After exiting the modal window, all concurrent threads must be re-entered in the states they were before. Modeling exclusive execution with Petri nets requires extensions of the formalism, as discussed for instance in [11].

Tokens that are specific to Petri nets can be used both in controlling and simulating system behavior, and in modeling data and resources of the system. If the place *Begin* of the Figure 6 contains only one token, the system can only execute one use case at a time. When the place contains n tokens, n concurrent executions of different use cases

are possible. It may even be possible to execute n scenarios of the same use case (multiple instances).

Table 3 summarizes the differences between Petri nets and Statecharts based on the above discussion, indicating the strengths (+ for good and ++ for very good) and weaknesses (-) of the two formalisms. We believe that the considered criteria are all highly relevant to modern UIs. Depending on the type of UI at hand, one or the other formalism and modeling approach will be more appropriate.

Criterion	Petri net	Statechart
Concurrency	++	+
Non-determinism	+	-
History states	-	+
Multiple instances	+	-

Table 3: Differences between Petri nets and Statecharts from a UI perspective.

5.4 Validation of Approach

The scenario integration and prototype generation activities have all been implemented in Java, resulting in about 4,000 commented lines of code. The Java code generated for the UI prototype is fully compatible with the interface builder of Visual Café. For obtaining CPNs in serialized form, as required by the scenario integration algorithm, the XML tools mentioned at the end of Section 3.2.2 are used.

Our approach has been successfully applied to a number of small-sized examples. For further validation, we have started implementing the suite of examples used for validating *SUIP* [22], the implementation of our Statechart-based approach [5]. This suite includes a library system, a gas station simulator, and a filing system. The gas station simulator will be of particular interest since it involves two actors. “Extreme” examples, i.e., examples that lend themselves particularly well to the Petri net or the Statechart based approach, respectively, will also be examined, in order to further investigate the differences between the two approaches (cf. previous subsection and discussion of system versus object views).

6 Conclusion and Future Work

The work presented in this paper proposes a new approach to the generation of UI prototypes from scenarios. Scenarios are acquired as SequenceDs enriched with UI information. These SequenceDs are transformed into CPN specifications from which the UI prototype of the system is generated. Both static and dynamic aspects of the UI are derived from the CPN specifications.

The most interesting features of our approach lie in the automation brought upon by the deployed algorithms, in the use of the scenario approach addressing not only sequential scenarios but also scenarios in the sense of the UML (which supports, for instance, concurrency in scenarios), and in the derivation of executable prototypes that are embedded in a UI builder environment for refinement. The obtained prototype can be used for scenario validation with end users and can be evolved towards the target application.

As future work, we plan to move in three directions. As mentioned above, we intend to further pursue our comparison of modeling approaches. This will also include the study of the interrelationship between modeling formalism and the UI paradigm being supported. Second, we wish to investigate backward engineering, that is, allowing the automatic modification of scenarios through the UI prototype. Finally, we plan to further study the verification aspect of scenario-based modeling [4], especially the completeness of the system specification obtained from partial descriptions in form of scenarios.

Acknowledgments

We would like to thank Mazen Fahmi for helping us in the implementation part of this work. Our thanks also go to our colleagues in the *Forspec* project, Gilbert Babin, Jules Desharnais, François Lustman, and Ali Mili.

Bibliography

- [1] J. Clark. SX: An SGML System Conforming to International Standard ISO 8879, <<http://www.jclark.com/sp/sx.htm>>.
- [2] K. W. Derr, Applying OMT: A practical step-by-step guide to using the Object Modeling Technique, SIGS BOOKS/Prentice Hall, 1996.
- [3] designCPN: version 3, Meta Software Corp. <<http://www.daimi.aau.dk/~designcpn>>.
- [4] M. Elkoutbi, User Interface Engineering based on Prototyping and Formal Methods. PhD thesis, Université de Montréal, Canada, April 2000. French title: Ingénierie des interfaces usagers à l'aide du prototypage et des méthodes formelles. To appear.
- [5] M. Elkoutbi, I. Khriiss, and R. K. Keller, Generating User Interface Prototypes from Scenarios, in Proceedings of the Fourth IEEE International Symposium on Requirements Engineering (RE'99), pages 150-158, Limerick, Ireland, June 1999.
- [6] M. Elkoutbi and R. K. Keller, Modeling Interactive Systems with Hierarchical Colored Petri Nets, In Proc. of 1998 Adv. Simulation Technologies Conf., pp. 432-437, Boston, MA, April 1998. Soc. for Comp. Simulation Intl. HPC98 Special session on Petri-Nets.
- [7] P. Hsia, J. Samuel, J. Gao, D. Kung, Y. Toyoshima, and C. Chen. Formal approach to scenario analysis, IEEE Software, (11)2, March 1994, pp. 33-41.
- [8] IBM, Systems Application Architecture: Common User Access – Guide to User Interface Design – Advanced Interface Design Reference, IBM, 1991.
- [9] IBM, XML Parser for Java, in Java Report's February 1999, <<http://www.alphaworks.ibm.com/formula/xml>>.
- [10] K. Jensen, Coloured Petri Nets, Basic concepts, Analysis methods and Pratical Use, Springer, 1995.
- [11] M. Kishinevsky, J. Cortadella, A. Kondratyev, L. Lavagno, A. Taubin, and A. Yakovlev. Coupling asynchrony and interrupts: Place chart nets. In P. Azema and G. Balbo, editors, ATPN'97, pp.328-347, Toulouse, France, June 1997. Springer-Verlag. LNCS 1248.
- [12] Koskimies K. and Makinen E.: Automatic Synthesis of State Machine from Trace Diagrams, Software Practice & Experience (1994), 643-658.
- [13] Ismail Khriiss, Mohammed Elkoutbi, and R. K. Keller. Automating the synthesis of UML statechart diagrams from multiple collaboration diagrams. In J. Bezivin and P. A. Muller, ed., <<UML>>'98: Beyond the Notation, pp.132-147. Springer, '99. LNCS 1618.
- [14] B. A. Nardi, The Use of Scenarios in Design, SIGCHI Bulletin, 24(4), October 1992.
- [15] P. Palanque and R. Bastide, Modeling clients and servers in the Web using Interactive Cooperative Objects, Formals Methods in HCI, Springer, 1997, pp.175-194.
- [16] C. Potts, K. Takahashi and A. Anton, Inquiry-Based Scenario Analysis of System Requirements, Technical Report GIT-CC-94/14, Georgia Institute of Technology, 1994.

- [17] F. de Rosis, S. Pizzutilo and B De Carolis: A tool to support specification and evaluation of context-customized interfaces. *SIGCHI Bulletin*, 28, 3, 1996.
- [18] M. B. Rosson. Integrating Development of Task and Object Models, *Communications of the ACM*, 42(1), January 1999, pp. 49-56.
- [19] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*, Addison Wesley, Inc., 1999.
- [20] E. Schlungbaum and T. Elwert, Modeling a Netscape-like browser Using TADEUS Dialogue graphs, In *Handout of CHI'96 Workshop on Formal Methods in Computer Human Interaction: Comparison, Benefits, Open Questions*, Vancouver, 1996, pp.19-24.
- [21] Siegfried Schönberger, Rudolf K. Keller, and Ismail Khriss. *Algorithmic Support for Model Transformation in Object-Oriented Software Development. Theory and Practice of Object Systems (TAPOS)*, 2000. John Wiley and Sons. to appear.
- [22] SUIP. Scenario-based user interface prototyping: Website and public domain software distribution, September 1999. <<http://www.iro.umontreal.ca/labs/gelo/suip/>>.
- [23] Symantec, Inc. *Visual Café for Java : User Guide*, Symantec, Inc., 1997.

Decidability of Properties of Timed-Arc Petri Nets

i u i¹ l n n l ui² n l u n l n¹
¹ i m n m i y g m i n m i
 ni i m l n i
 defrutos,alonso @sip.ucm.es
² n m i li ni
 ni ill n l
 valentin@info-ab.uclm.es

Abstract. im in n ing l
 in i l y nn im l n i z ing
 l in i m l n in igni n ly
 x i n n im in i i n in i
 n iff n n m ig n g
 m ili y l m n i l n n
 ili y n n n i l i i
 n n l in l i n i n n n
 n i imil l n n ly in lly
 i n n nn ing
 ny n i i n in n in n in n
 ff i y

1 Introduction

in i l u lin n n l i n u n
 u i i l n u n li i l un in u
 in l i n i n i l n
 n i li in i i i n n
 i u ul ni in lin n n l i
 u i i u uni i n n l i
 u i n in u i in i n i
 n in n i n i u l i i n i
 l n i i n i u in n ini i lu l l
 in i ili i i u i n l u i in l
 li n lin i n i i n l in ll l l
 in u i n l n u n l i in
 i n l il l n un il l n il l n
 n i i l u in n i i n il un il l n
 nn i in i i n
 il l l u i i l i l n in un il l
 u n i n l n n i l ini

i ig l n n l iz n lg n l n

n n l l i ll i in i in
n in in l i ll i i l n i i n n
u l i i in n in nl l n i i n
n in i ul u l i n l
n i nl i in l i n i n
i n in i n i n u l¹ lu in i in
l i i i n i n l n i i n
l l ll i in l i li i i n n
n n u j n n i i n
n l n i . i i in in i
lin l in i n n nl i n u n
in i n n u i in n in
ll i n in in i in n u
ni u i l l in n ni i n
n l i n u i u i n
n n u in i n l l
ul n i i n i in l n i i n l il n ni
i n i n ul l n i i n l i n
i n in l i u
i n i in l li i in i i n l
i ul in n i l l in ui
n n u n in n ni i n i n n n in i
in i n n n ni n i i n l
n li i n i in i i
ui i in lin n i
n i i n l in i ul
in l n ju i i l l i n n
n l i n u n
n i i i in n u in l in in
i ul nn l i ul un in u ul
i n i un i i n in i n
ul n i ni n l n in
i n i i n u ili l un i l i
i n
n i n u i ili i
in in i ul ili n un n i l
i i n u n l nn i n n n n
i n i ul in i n u n

¹ l g i i l n ing l n m m g
im in i ill nly ll n l n m in im
ing n n l g n liz i ili y l n in
im m l n n i g n l n ing n i ili y
 ili y in i i im n n x m l ing n
i ili y m n n in im ili y i l n i l n
i im i n i

i ili y i im i
 un i ili n i ul i n n n l
 n l
 n n n n u l u i in l ll
 i ul ll n li l i in n l n
 n i ul i i ili i i l l
 n l in n n l² ni u n in l n
 n un n i i l n u n n ll
 i i ili ul i l un n i un i l in
 u in i ili l l i i
 ill n n n in n u i i in in
 n n l i u u n in
 l n n i i n in u u u i
 l in n in n n in il l ill
 i in n n i l u
 ili n i i n i l l l i ili i i
 i l
 i u u ll n i n n i i
 n n i n i in in ll un i ili il
 i n in i i l i u in
 n i u i n n i n i ili ili
 i n i in in l l i n i
 n n n n n n un n
 n n un i ili ul n n n i n
 l in n i i l n l u
 i n in ll in i n i u
 n in u u

2 Timed-Arc Petri Nets

l i i in i i n nn i n
 n in lu in i in l i i in n
 nn in l i n i i n i i in l i
 li i n n u j n n i i n
 n i i n i n n ll i n i i n
 n in n i n u n i u n i
 n n i u i in n l in l i
 n n in u n i n in in n li
 n i i n ill nl n n l n ui i
 u n i n l n l ili
 n i i n i n n l n i i i l
 n i n ni l u l u in in

² i m inly n in i i g n li y i m m
 n n l n li ny ll n i i n y m
 n *backward* lg i m i ili y n n in
 l ng g in

i ig l n n l iz n lg n l n

i ill n ll in u u in n i n i i n
n i i n

³ N P, T, F, times
P T ∅
F fl F P T T P times
 p, t
. . times F P T - .

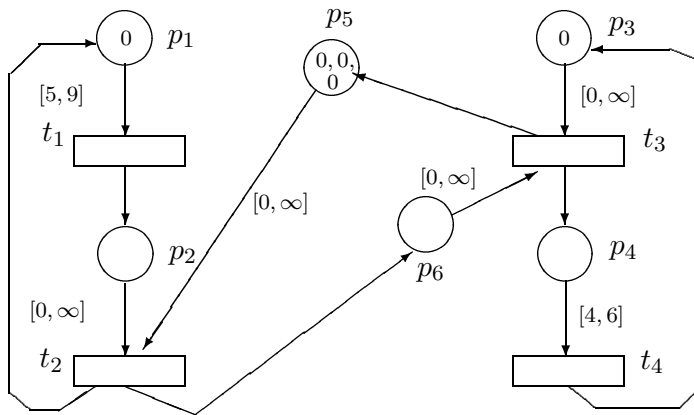
n times p, t t₁, t₂ i π_i p, t n t_i i 1, in
times n in l n n u in
n i i n ill l π₁ p, t π₂ p, t
ill i x times p, t n π₁ p, t x π₂ p, t

i u l n i n n nn i n u l nu
in n n ul i n l in M
i un in M P - n ni ul i
n u l nu u u u l l i nn i in nu
n u n i n u l nu i
ill n in N N n u in l i l n in
n l nu n n l p M p in ll
u n i n i ill n in i i u l n in in
M i p, i i p P n i n in i l i in p, i M
ini i l in nl ll in M u ll p in P n
n x > M p x i n n i ⁴ n
n i N, M N i
i i n n M i n ini i l in n i u u l i
ini i l in ill in n in n l i in
n i i n i n n u n i n n n in
ni n n n i i n ll in i in n
in ni ili in u n in
i i n i n i in n i ll
n n in u u l n i n n i n in
i in i n n i ul ill u n n
i ini i ll ni ll i n lu l llin
l n i l lin u n u
u n n n i i n n l
i

N P, T, F, times M t T.

³ im li y m ni i n n i nly i ig x n i n
g n l i g ig i ig
⁴ n i l n l m ll ini i l m ing n ining l n
l ng nly ni n m n n m in n im ing
i i i n i in i i i ini i l n n y
im m l

i ili y i im i



i i n lin l

$$\begin{array}{c}
 t \\
 p \bullet t \ x_p \qquad M \ p \ x_p > \ x_p \ \text{times } p, t \\
 \dots \\
 \text{times } p, t . \\
 t \qquad M \\
 M
 \end{array}$$

$$\begin{array}{l}
 M \ p \quad M \ p - C^- \ p, t \quad C^+ \ t, p \ , \ p \ P \\
 C^- \ p, t \quad \left\{ \begin{array}{l} x_p \ x_p \ M \ p \ n \ x_p \ \text{times } p, t \ i \ p \bullet t \\ \emptyset \end{array} \right. \\
 C^+ \ t, p \quad \left\{ \begin{array}{l} \emptyset \quad p \ t \bullet \end{array} \right. \\
 t \\
 t.
 \end{array}$$

$$\begin{array}{c}
 M \ t \ M \\
 t
 \end{array}$$

age

$$\begin{array}{c}
 \text{age} \quad N \quad - \\
 \text{age } M, x \ p \ y \quad \left\{ \begin{array}{l} M \ p \ y - x \quad y \ x \\ M \ x \\ \text{age } M, x . \end{array} \right.
 \end{array}$$

i ig l n n l iz n lg n l n
 l u n lu i n in in l n i i n i n
 il n in n i i n in n i
 i n ul in in l ul l in u n
 in n u n i u in n i i n n i
 in n i i n n M R M l n in
 n i i n n i n n i n i
 i u n u n σ $M_0 R_1 x_1 M_1 \dots M_{n-1} R_n x_n M_n$
 $M_i s$ in $R_i s$ ul i n i i n n $x_i s$ in u
 $M_i R_{i+1} M_{i+1} n M_{i+1}$ age M_{i+1}, x_{i+1} i x_i in
 ll null in l i n u ll l
 n i n i l ul in in l
 in in i in i n n ll li n u n i l
 i in ll n u n σ $M_0 \overline{R}_x M_n$ \overline{R} $R_1 R_2 \dots R_n$ n
 $x \sum_{i=1}^n x_i$ n n n lu x_i
 in in i in l n
 i n l i n i u n i n l u in
 in u i n i n n in i
 i n i l n i i
 i l n l l
 i n N, M_0 n M_0 l in
 n N in M_0 i n m n $M_0 m$ l
 in u i n l u i n m
 $M_0 m$ M $\overline{R}, M_0 \overline{R}_m M$
 n M $M_0 m$ M i l in i m N, M_0
 i i n in ni u i n l p P i i
 i n u ll M M_0 M p n i i
 i ll t i n u ll M M_0
 $M p t$ n n ll i i i i
 n u ll t n ll M M_0 M p t n
 N, M_0 i
 i p P in i i un i l ll un uni l
 i l ll un ⁵
 n l n in nl n uni l i l ll un
 i i i i ll in

$$N \qquad N, M_0 \qquad p \quad P \qquad p$$

. in nl n ll M_1 M_0 N n ll m
 i M_2 M_0 N u ll p P M_1 p m M_2 p

⁵ nn n n n i l n ly y ying ili y
 i ni in y ging m ing n l y in l n l iff n
 m ing

i ili y i im i

n u n i p i i l ll un in N n in un
i uni un n

3 Decidability of Timed Reachability

n ll in ul

i ili i i l in i n ni i m n n
i ul i u u in n n l in i
n N^m i i n ll

N P, T, F, times M_0
6 m N^m P^m, T^m, F^m

P^m P $..m$ c_i i $..m$

T^m t, ages, l ages p p, t F times p, t $..m$ $, l$ $..m$
 tick_i i $1..m$

f_l F^m P^m T^m T^m P^m

F^m c_{i-1}, tick_i $, \text{tick}_i, c_i$ i $1..m$
 c_i, t, ages, i i $..m$
 p, j $, t, \text{ages}, i$ p, t F, j $i - \text{ages}$ p
 t, ages, i $, p, i$ t, p F, a $..i$

M_0^m

$$\begin{cases} M_0^m c_0 & 1, \\ M_0^m c_i & i > , \\ M_0^m p, & M_0 p \\ M_0^m p, i & p P, \\ & p P \quad i > . \end{cases}$$

i n i ul N^m i ul in i u N u
in n m i i n i n ll in i
n n n l in c_i
in n in n lu l l l n
l n i i n ul n u n i n l n i
u u n n n l p, j P^m i
u l $i - j$ c_i i u n l l l ll i i li
ll

6 l n i l x n i n i n in ll ny ni
m ing ini i l m ing M_0 n P^m P -old M_0 $..m$
old i n in n p, i in M_0 l n y
n n n ing l $p, -i$ P^m

$$\begin{aligned}
 & \text{ig} \quad \text{l n n} \quad \text{l} \quad \text{iz n} \quad \text{lg} \quad \text{n l n} \\
 & N \quad P, T, F, \text{times} \quad M_0. \\
 & M \quad N \quad m \\
 & \text{old } M \quad \max \quad i \quad i \quad M \quad p \quad . \\
 & \text{old } M \quad m \quad M^m \quad N^m \\
 & \begin{cases} M^m \quad c_m \quad 1, \\ M^m \quad c_i \\ M^m \quad p, j \quad M \quad p \quad m - j \end{cases} \quad \begin{matrix} i < m, \\ j \quad ..m \end{matrix} \\
 & M \quad N^m. \\
 & M \quad M_0 \quad m \quad M^m \quad M_0^m \quad N^m. \quad N \quad \text{old } M \quad m
 \end{aligned}$$

4 Decidability of Coverability

$$\begin{aligned}
 & \text{n} \quad \text{i} \quad \text{i n} \quad \text{ill} \quad \text{ili} \quad \text{i} \quad \text{i} \quad \text{l} \quad \text{i} \quad \text{i} \\
 & \text{n} \quad \text{l} \quad \text{i} \quad \text{ul} \quad \text{nn} \quad \text{in} \quad \text{n} \quad \text{li in} \\
 & \text{ili} \quad \text{l in} \quad \text{n} \quad \text{u} \\
 & \text{ni u} \quad \text{n l} \quad \text{n in} \quad \text{l} \\
 & S \quad Q, \quad , \\
 & Q \quad M, \dots \\
 & \quad Q \quad Q \\
 & \quad Q \quad Q \quad q \\
 & fl \\
 & \quad M \quad M \quad M_1 \quad M \quad M_1 \quad M_1 \quad M_1 \quad M. \\
 & S \quad Q \quad . \\
 & M \quad K \quad K \quad Q \quad M \quad M. \quad M \quad K \\
 & \quad K \quad Q \quad s \quad \text{cover}_s \quad K \\
 & \quad K \quad B \quad K \quad B \quad K \\
 & K \quad M \quad M \quad B, M \quad M. \\
 & \quad \text{n} \quad \text{n} \quad K \quad Q \quad K \quad \text{n} \quad M \quad Q \quad M \quad K, M \quad M
 \end{aligned}$$

S

S . S

$\text{cover}_1 B$. $B Q$

ll in i i ul in i
n

– M n M_0 i M M_0 i M M n
n i l u i n in M M_0 u
 M t M $\text{cover}_1 M$ in M

– in i n u i n in $\text{cover}_k M$ ll k n
n i u l

n n $\text{cover}_k M$ $\bigcup_l \text{cover}_l M$ i u l k k i li

$\text{cover}_k M$ $\text{cover}_{k'} M$ u l in i ll
u i in u i k u $\text{cover}_{k+1} M$

$\text{cover}_k M$ $\text{cover} M$ u i i l M n
 M_0 i M $\text{cover} M_0$

– in $\text{cover}_k M_0$ in ni ul n i l i
in n i u l in ll n i l i
i ni i

n l i ni u i ili ili
l n n u in l in n

i in n u l n i ul ul i in lu in i i
i i n li in u in in n n

un l ul n in i n in ni n ni n l
 p n i in l n in M_i p, i $M_i \not\subseteq M_j$ n

i j l i i i l n n u in in in
un n n l n u i uni

n in i n nl u n i n i M_2 p, t
i i li ll

N P, T, F, times .

p P

p $\pi_1 p, t$ t p^\bullet $\pi_2 p, t$ t p^\bullet , $\pi_2 p, t <$
 $S p$ p 1.

N $St N$ $S p$ p P .

i ig l n n l iz n lg n l n
 u n n n p p nl
 n i i n n i i n t p^\bullet ul u in n
 i π_2 p, t i u l in u n i i n t
 in l n n p i uni n n i S p i u
 in uni ll l n i St N in
 lu S p
 n n n u in l i n n in
 i in ll

$$N \quad P, T, F, \text{times} \quad M \, M$$

$$. \, M \quad M$$

$$M_{St(N)-1} \quad M_{St(N)-1} \quad p \quad P \quad i \quad M$$

$$k \quad M_k \quad p \quad i \quad \left\{ \begin{array}{l} M \quad p \quad i \quad i \quad k \\ St \, N \quad i \quad M \quad p \quad i \quad St \, N \end{array} \right.$$

$$q$$

$$M_1 \quad M_2 \quad t \quad T \quad M_1 \, t \quad M_2 \, t \quad M_1 \, t \, M_1$$

$$M_2 \quad M_2 \, t \, M_2 \quad M_1 \quad M_2.$$

$$n \, i \quad i \quad n \quad u \, n \quad ll \quad in$$

$$N$$

$$.$$

$$.$$

$$.$$

$$1 \quad i \quad li \quad i \quad n \quad in \quad ul$$

$$M \quad in \quad n \quad n \quad i \quad i \quad i \quad l \quad M_0$$

$$M \quad M_{St(N)-1} \quad n \quad n \quad l \quad u \quad in \quad 1 \quad in$$

$$i \quad i \quad i \quad i \quad l \quad i \quad i \quad u \quad nn \quad i \quad l$$

$$u \quad u \quad n \quad in \quad i \quad n \quad u$$

$$n \quad n \quad l \quad n \quad St \, N \quad l$$

$$in \quad in \quad ni \quad n \quad n \quad i \quad M \quad i \quad p, i \quad M <$$

$$in \quad in \quad l \quad i \quad n \quad i \quad n \quad l \quad u \quad l \quad in$$

$$St \, N \quad M \quad 1 > St \, N \quad n$$

$$i \quad n \quad in \quad i \quad i \quad n \quad i \quad ull \quad in \quad i \quad n \quad u$$

$$n \quad n \quad un \quad n \quad M \quad u \quad in \quad i$$

$$in \quad l \quad i \quad n \quad n \quad n \quad i \quad i \quad i \quad n \quad in \quad M \quad i \quad l$$

$$M_0$$

i ili y i im i
 ul n inu u i i i n
 in i u in i i i n i il nin
 in l n u in n n ill in
 n i n n n i i n n i ul
 i i l n l n n ul u
 i ili n un i ili n n

5 Relating Timed-Arc Petri Nets and Transfer Nets

i i n n in n n i ul
 n li l i in n i n u l u i in
 1

$$\begin{aligned}
 & N \quad P, T, F \\
 & P \quad p_1, \dots, p_P \\
 & T \quad P \quad T \quad \emptyset \\
 & F \quad P \quad T \quad T \quad P \quad \mathbf{fl} \\
 & \quad t, p \quad T \quad P, F \quad t, p \quad . \\
 & \quad F \quad p, t \quad p, t \quad t \quad p. \\
 & m_0 \quad P \quad N, m_0 \quad N \\
 & \quad N \quad P, T, F, TA \quad P, T, F \\
 & \quad TA \quad TA \quad T \quad P \quad P \\
 & p, p \quad TA \quad t \quad \left(\quad p \quad p \quad p, p \quad TA \quad t \quad F \quad p, t \quad F \quad t, p \quad \right). \\
 & m_0 \quad P \quad N, m_0 \quad N \\
 & \quad i in \quad ul \quad n \quad n \quad n \quad n \quad ll \\
 & \quad t \\
 & \quad t \quad M \quad M \\
 & \quad F \quad p, t \quad M \quad p \quad F \quad t, p \quad p \quad t.
 \end{aligned}$$

i ili y i im i

$$\begin{array}{c} M \quad N \\ \left\{ \begin{array}{l} M^B \quad p, a \\ M^B \quad p, B \end{array} \right. \quad M \quad p \quad a \quad a < B \\ \sum_a \quad B \quad M \quad p \quad a \end{array}$$

$$M^B \quad N^B$$

$$M \quad tr \quad M \quad tr \quad M \quad M \quad N \quad M^B \quad M \quad .$$

$$u \quad ll \quad in \quad li \quad N^B \quad i \quad ull \quad i \quad ul \quad i \quad n \quad N$$

$$B$$

$$N \quad B \quad St \quad N \quad N^B$$

$$\begin{array}{c} M \quad N^B \quad M_1, M_2 \quad M \quad M_1 \quad B-1 \quad M_2 \quad B-1. \\ M_1 \quad t \quad N M_2 \quad t, \text{ages} \quad T^B \\ M_1^B \quad t, \text{ages} \quad N^B M_2^B. \\ M_1 \quad t, \text{ages} \quad N^B M_2 \quad M_1 \quad M_1 \quad M_1 \quad t \quad N M_2 \\ M_2 \quad M_2 \quad . \end{array}$$

$$\begin{array}{c} u \quad n \quad n \quad i \quad l \quad n \quad l \quad i \quad ul \quad n \quad n \quad n \\ l \quad i \quad l \quad n \quad i \quad ul \quad ul \quad u \quad i \quad i \quad ul \quad i \quad n \\ i \quad ili \quad ili \quad u \quad i \quad in \quad i \quad in \\ i \quad l \quad un \quad St \quad N \quad i \quad ll \quad u \quad n \quad u \\ n \quad in \quad ni \quad l \quad ll \quad in \end{array}$$

.

.

$$\begin{array}{c} i \quad n \quad l \quad in \quad i \quad i \quad ul \quad i \quad n \quad n \quad l \quad n \quad lu \quad i \quad ili \quad in \\ i \quad n \quad l \quad i \quad nn \quad n \quad in \quad i \quad in \\ u \quad i \quad n \quad n \quad N^B \quad n \quad in \quad u \quad tick \quad n \quad i \quad in \\ n \quad n \quad i \quad in \quad n \quad i \quad n \quad u \quad n \\ u \quad i \quad n \quad in \quad ni \quad u \quad i \quad n \quad i \quad n \quad in \quad i \quad n \quad in \quad ni \\ u \quad n \quad in \quad in \quad nl \quad tick \quad n \quad i \quad in \quad u \quad u \quad u \quad u \quad n \\ n \quad i \quad u \quad in \quad tick \quad l \quad St \quad N \quad i \\ l \quad in \quad ll \quad n \quad St \quad N \quad l \quad i \quad i \quad i \\ i \quad n \quad n \quad nl \quad in \quad i \quad i \quad n \quad ul \\ i \quad u \quad n \quad in \quad u \quad i \quad n \quad i \quad u \quad n \quad n \quad tick \\ n \quad i \quad in \quad in \quad n \quad l \quad in \quad ni \quad ili \quad ni \quad u \quad in \\ l \quad n \quad n \quad in \quad in \quad un \quad u \quad in \quad tick \quad ni \quad n \quad n \\ u \quad n \quad in \quad n \quad ll \quad in \end{array}$$

.

$$\begin{array}{c}
\begin{array}{ccccccc}
& i & & ig & l & n & n & l & & iz & n & lg & & n & l & n \\
n & & & & un & i & ili & ul & & n & n & & l \\
& n & & n & & n & i & ul & i & n & i & i & u & & l \\
& n & u & i & n & i & & n & i & ul & i & n & un & & in \\
& u & in & & un & i & ili & & ili & & & & n \\
in & 1 & 11 & & un & i & ili & un & n & & n & & in & i \\
& i & i & ul & i & n & in & u & n & i & i & n & n & n & n \\
& & n & i & i & n & n & u & n & n & in & n & l & i \\
& u & i & n & l & & u & & u & i & n & ill & n \\
i & & l & & n
\end{array} \\
\begin{array}{c}
\begin{array}{c}
N \\
P^I, T^I, F^I, \text{times}^I
\end{array}
\begin{array}{c}
P, T, F, TA \\
P^I, T^I, F^I, \text{times}^I
\end{array}
\end{array}
\begin{array}{c}
P^I \quad P \quad n \quad r_t \quad t \quad T \quad n \quad r_t \\
T^I \quad T \quad v_{t,p} \quad p \quad P, t \quad T \quad rn_t \quad t \quad T \quad v_{t,p} \\
t \quad p, p \quad TA \quad t
\end{array}
\begin{array}{c}
rn_t \\
f^l \quad F^I \quad F \quad F^I \quad P^I \quad T^I \quad T^I \quad P^I \\
p, v_{t,p}, r_t, v_{t,p}, v_{t,p}, r_t \quad p \quad P, t \quad T \\
v_{t,p}, p \quad p \quad P, t \quad T, p, p \quad TA \quad t \\
v_{t,p}, p \quad p \quad P, t \quad T, p, p \quad TA \quad t \\
r_t, rn_t, rn_t, n \quad t \quad T \\
n, t, t, r_t \quad t \quad T
\end{array}
\begin{array}{c}
\text{times}^I \quad F^I \quad P^I \quad T^I \\
\text{times}^I \quad p^I, t^I \quad \left\{ \begin{array}{c} p^I \quad r_t \quad t \quad T \\ 1, 1 \end{array} \right. \\
i \quad N^I \quad n \quad i \quad ul \quad i \quad u \quad n \quad N \quad in \quad ll \quad in \\
in \quad M \quad N \quad ill \quad n \quad in \quad M^I \quad i \quad n \\
p \quad P \quad i \quad M \quad p \quad k \quad M^I \quad p \quad k \quad k \quad n \\
ul \quad i \quad n \quad in \quad k \quad lu \\
M^I \quad n \quad ^1 \quad n \quad M^I \quad r_t \quad \emptyset \quad ll \quad t \quad T \\
n \quad n \quad M \quad t \quad _N M \quad n \quad n \quad in \quad N^I \quad in \quad t \\
n \quad ll \quad in \\
M^I \quad \emptyset \quad _1 \text{ages} \quad M^I, 1 \quad i \quad n \quad uni \quad ll \quad n \quad in \quad M^I \\
\text{ages} \quad M^I, 1 \quad t \quad _0 M^I \quad i \quad t \quad n \quad u \quad n \\
n \quad in \quad \bullet t \quad l \quad in \quad 1 \quad il \quad n \\
in \quad t \bullet \quad l \quad in \\
n \quad in \quad in \quad n \quad in \quad M \quad in \quad N^I \quad M^I \\
n \quad n \quad in \quad p \quad P \quad i \quad i \quad p, p \quad TA \quad t \\
n \quad in \quad p \quad i \quad i \quad in \quad M^I \quad p \quad 1 \quad i \quad v_{t,p} \\
i \quad p \quad P \quad i \quad i \quad n \quad n \quad p, p \quad TA \quad t \\
ju \quad n \quad n \quad in \quad M^I \quad p \quad in \quad M^I \quad p \quad 1 \quad i \\
n \quad in \quad n \quad i \quad i \quad n \quad v_{t,p} \quad l \quad N^I \quad in \quad rn_t
\end{array}
\end{array}$$

i ili y i im i

 i n n i i n n un il in
 i ul i n in l n i i n rn_t nl
 i ili i i i l ll u u inin
 M^I rn_t in n in M^I n i l i
 n M^I u in M^I n i u n in
 M^I M^I l n in M^I in l
 in nl in i N^I n l i in M^I n in u
 n l u in u u in n n i i n
 in N^I n n u n l n l
 in ll in

 N^I N
 N^I

 N, M_0 M^I M^I .

 n u n un i ili ul n n n
 n l n i ul ll in

 .

 . in j i n in i ul in n N^I n
 n in ju n i n l in in
 N n i i i n p P p i un in N i
 i k u ll M M^I N^I M p k
 in l un n i un i l n n n lu
 i l ll un n i l un i l

 n ll in i n ill n n u n u u l
 u i n n n n n ill n
 n u i n n u i ul n nl in lu in in
 n i i n $v_{t,p}$ l p P i F p, t u in i
 n n l li i u i in l n in i
 l n l l l un n n i i u
 n l u l u in in
 in n in n ju i i ju n i n
 n u n influ n in in i u
 n u in in l in n u l l
 un n n n u n i n
 n i l un n i n i l

6 Eliminating Dead Tokens

u n u n u n i n i i u n li i
 n i u in u u i u n l u ili
 ill n in i l in i
 n l i in in in n in

$$\begin{array}{c}
\begin{array}{ccccccc}
& i & & ig & l & n & n & l & & iz & n & lg & & n & l & n \\
& & in & & & l & & i & & l & & n & & in & & in \\
li & in & in & & n & & u & & in & & & & nu & & & \\
l & & in & & u & li & l & & u & in & & l & i & n & & \\
& l & & n & & & & & ni & & & & & & &
\end{array} \\
\\
\begin{array}{ccccccc}
M & & N & P, T, F, \text{times} & & M & \\
& t & & M & & M & M & & t & T & \\
& & ill & & n & u & n & & i & ili & & ili & & in \\
& & & n & n & & i & l & & & i & i & n & & in \\
n & i & & i & ll & n & & & li & l & n & & in & & \\
& & n & n & & in & l & & & & & & & &
\end{array} \\
\\
\begin{array}{ccccccc}
& & & & N^{CB} & & N \\
& & & N & & & \\
B & N^{CB} & P^B & \text{clock}, T^B, F^{CB}, TA^B & & F^{CB} & F^B \\
\text{tick}, \text{clock} & . & & & & & \\
M_1 \text{ clock} & M_2 \text{ clock} & & M_1 \text{ }_{PB} & M_2 \text{ }_{PB} . & N^{CB} & M_1 \text{ }_B M_2
\end{array} \\
\\
\begin{array}{ccccccc}
& n & in & i & & in & in & B & in & n & \\
n & l & & i & n & i & i & l & l & l & i & \text{tick} & n & & in \\
n & u & ul & i & & u & i & i & l & & B & i & n & ll & u & i & in \\
nu & & \text{tick} & n & i & n & un & & i & i & un & l & i & i & nl & n & \\
& u & i & u & & n & n & un & i & n & & n & li & l & n & in & \\
ni & i & n & N^{CB} & & & & & & & & & & & &
\end{array} \\
\\
\begin{array}{ccccccc}
& B & & N^{B, B'} & & N^{CB} & \\
P^{B, B'} & P^B & c_0, \dots, c_{B'} & & & & \\
T^{B, B'} & T^B & \text{tick}_1, \dots, \text{tick}_{B'}, \text{tick} & & & & \\
F^{B, B'} & F^B & c_{i-1}, \text{tick}_i, \text{tick}_i, c_i \text{ }_i \text{ }_{1..B} & & & & \\
& & c_{B'}, \text{tick} \text{ }_i, \text{tick} \text{ }_i, c_{B'} . & & & &
\end{array} \\
\\
\begin{array}{ccccccc}
& & B, B' & & M_1 & B, B' & M_2 \\
M_1 \text{ }_{PB} & M_2 \text{ }_{PB} & & M_1 \text{ }_{PB, B' - PB} & M_2 \text{ }_{PB, B' - PB} & & \\
& n & & & in & M & in & \sum_{i=0}^{B'} M \text{ }_i c_i \\
1 & i & i & l & & i & ini & i & l & in & n & M_0 & i & M_0 & c_0 & 1 & n \\
M_0 & c_i & , & i & ll & & l & in & l & & n & i & & i & & \\
& B, B' & & l & & in & i & i & & & n & in & & & & \\
& n & i & i & n & & i & ll & u & u & & & & & & \\
& u & n & & n & i & & n & p, i & in & i & n & in & & N & & \\
& i & ini & i & l & in & M_0 & i & i & n & & n & & & & l &
\end{array}
\end{array}$$

$$\begin{array}{c}
\text{in } M_0 \overline{R}_m M \\
t \ M \ t \ i \ p, t \ F \ n \ i \ m \ \text{times } p, t \\
\text{ll in}
\end{array}
\begin{array}{c}
\text{ili y} \\
\text{times } p, t \\
p, i \\
\text{times } p, t \\
M \\
M_0^{B, B'} \\
M_0^{B, B'} c_0
\end{array}
\begin{array}{c}
\text{im} \\
\text{ll} \\
\text{times } p, t \\
M_0 \\
\text{times } p, t \\
N^{St(N), St(N)} \\
M \\
M_0^{B, B'} \\
c_0
\end{array}
\begin{array}{c}
\text{i} \\
\text{n} \\
\text{n} \\
M_0 \\
\text{times } p, t \\
M \\
M_0^{B, B'} \\
1
\end{array}
\begin{array}{c}
\text{i} \\
\text{n} \\
\text{n} \\
M_0 \overline{R}_m M \\
\text{times } p, t \\
M \\
M_0^{B, B'} \\
1
\end{array}$$

$$M \quad N^{St(N), St(N)} \quad M \quad t \quad n \quad M \quad c_m \quad 1 \quad i \quad i \quad m \quad \text{times } p, t$$

$$in \quad u \quad i \quad n \quad n \quad i \quad n \quad i \quad u \quad l \quad in \quad nn \quad u \quad n \quad n$$

$$n \quad i \quad i \quad n \quad in \quad u \quad u \quad n \quad l \quad i \quad n \quad i \quad l \quad n \quad in$$

$$l \quad i \quad i \quad i \quad l \quad u \quad n \quad ll \quad n \quad u \quad u$$

$$n \quad i \quad i \quad n \quad i \quad un \quad u \quad n \quad u \quad n \quad ul$$

$$i \quad n \quad i \quad l \quad n \quad i \quad in \quad l \quad n \quad i \quad n \quad i \quad n \quad n$$

$$M \quad M \quad p \quad 8 \quad P, T, F, \text{times}$$

$$M \quad M \quad M \quad p \quad M$$

$$9 \quad M$$

l i l l n in u n u

ll in ul

.

.

.

l u ili in n u

ili in l i n l i i in i il

l i n

⁸ n i n in n g i y in m l ni i n

⁹ ing in n gi n n i ly g n ing i

m i n

i ig l n n l iz n lg n l n

 n i n n N^B i ul in N n in
 l u in u in in i l l in l n i
 n n l n l n in
 i n n i n u in n i i n n
 in u un l u n i n
 n i i l i n n i l i n u in i
 un l l n n in i n in
 l in i in l n in ili i
 i l n lu in i l n
 i l l

n i u i n i i n n in n i n
 n i n n n i i n
 i n n i i n in n n n li i
 n u in i n n ill n
 n n i i n n n i ul n i
 i n n in i u in i i l n l
 n l l n i n n i n u

7 Conclusions, Discussion, and Future Work

 u i i in i ili
 in in n n un n i l u i l ll un n i
 n l n n i l ll i i
 u l l in i n n n n u n lin
 n i i n i l l l ili
 i l un n i i
 in i in u un un l n n i
 i l n in n ul N i i i n
 in ni u i n i ni u i n i u nu in
 in i i ni
 in un l i n i n n n n
 in i i in i ul l i
 n n n i li

 i in i n i l l in n u
 ni u i i i in ni u i n n l in
 n i nn n n in ili n
 i n u in i n i u u n
 in n u n l u i ili i
 ui l n l n n n u i l i
 i l in l i i l i in n n
 in l i ul in un n in

i ili ul n u ni u in 1
in u in ili
n nin i ili n in u ul n inu u i
nj u i n n in li i in l nin
n n u in n i i n in
i n l lu u ll in i i l n u i n
in i u ul ul in un
n i nl n in n i i i u n
ul n i u n ul n
q n in n in lu
i nj u i ul l n inui
u n n i ul i n l ul n
i in in l n l lu
l u n n i n i ili ili
i i n i n l n l l l u i
li nn i l nl i in ul
u n li i n ul n i l in i
in l n l n ul i i
nn n l n i in u n l n i
n ul u n li i n

References

- n l *Interval Timed Coloured Petri Nets and their Analysis*
l 5
ll n n n i n *General Decidability
Theorems for Infinite-State Systems* ym gi in m
i n n i ly g
m n n l i i l n n m ni
On Petri Nets with Stochastic Timing n n in l
n im i m i y 5
n l n x *Vector Addition Systems and Semi-Dyck Language*
i l l l ni i i n
ni m
5 l gn i n m n *The Weakness of Some Timed Models for
Concurrent Systems* ni l
l gn i i i n igil *From Timed Petri Nets to Timed LO-
TOS* n n in l ym i m n l
i i n ing n i i n ll n
n *Modelling Time in Petri Nets* n li n
n i l
n n ggi l ini *Time-based Expressivity of Time Petri
Nets for System Specification* i l m i n
5

i ig l n n l iz n lg n l n
 in l n n l n *Reset Nets between Decidability
 and Undecidability* 5 n ll m ng g n g m
 ming l g nm ly l 5
 ing l g
*Réseaux de Petri avec Reset/Transfert: Décidabilité et
 Indécidabilité* n i n l l m l i
 n
 n n n l n *Boundedness of Reset P/T Nets*
 n ll m ng g n g mming
 g z ly l ing l g
 in l *Reduction and Covering of Infinite Reachability Trees* n m i n
 n m i n
 in l n n l n *Fundamental Structures in Well-Structured
 Infinite Transition Systems* in m i n i l n m
 i ym i m m in zil l
 ing l g
 ni *Analysis of Place/Transition Nets with Timed-Arcs and its
 Application to Batch Process Control* li i n n y i
 l ing l g
 5 y *Lossy Counter Machines* ni l
 "n n
 lin *A Study of the Recoverability of Communication Protocols*
 i ni li ni
 n *Petri net Theory, and the Modeling of Systems* n i ll
 m z n *A Timed Calculus for LOTOS*
 5
 m n ni *Performance Evaluation of Asynchronous Concurrent Sys-
 tems by Timed Petri Nets* i n i n l
 gy m i g
 i i *Use of Petri Nets for Performance Evaluation* i
 n n i n l ym i m ing lling n l
 ing m y m l i i n li 5
 l ig n *Decidability of the Strict
 Reachability Problem for TPN's with Rational and Real Durations* 5
 n n i n l n i n m n l 5 5
 l ig n *On Non-Decidability of Reach-
 ability for Timed-Arc Petri Nets* n n i n l n i
 n m n l
 l *Timed Petri-Nets for Modelling and Analysing Protocols with Real-
 Time Characteristics* n l i i n
 ing n i i n ll n

Analysing the WAP Class 2 Wireless Transaction Protocol Using Coloured Petri Nets

n n n n ll n n

Cooperative Research Centre for Satellite Systems,
University of South Australia,
Mawson Lakes SA 5095, Australia
{sgordon,jb}@spri.levels.unisa.edu.au

Abstract. Coloured Petri nets (CPNs) are used to specify and analyse the Class 2 Wireless Transaction Protocol (WTP). The protocol provides a reliable request/response service to the Session layer in the Wireless Application Protocol (WAP) architecture. When only a single transaction is considered occurrence graph and language analysis reveals 3 inconsistencies between the protocol and service specification: (1) the initiator user can receive two TR-Invoke.cnf primitives; (2) turning User Acknowledgement on doesn't always provide the User Acknowledgement service; and (3) a transaction can be aborted without the responder user being notified. Based on the modelling and analysis, changes to WTP have been recommended to the WAP ForumSM.

1 Introduction

n n n n n n
n l l n l l
n n n n n
l n ll n l n n n
l n n n l
n n l l n n n n
n n n n l n l l n
l n l n x n
l l n l n l

1.1 Wireless Application Protocol

l l n l n n
n n n n n
l n n n l n x l
l n n n l n
n k n n l n
l n k l n n

SM n n n l
n n n n n n n
n l n l
n n n ll x n n l
n l n l n
n n n n l n n k n
n n n l n l l
n n k n k n
l l n x n l n n l
n l n n l
l k ll l l k l
n l n n n n l n l l
l n n n n n l k n
n n n n n l n
l l n n l

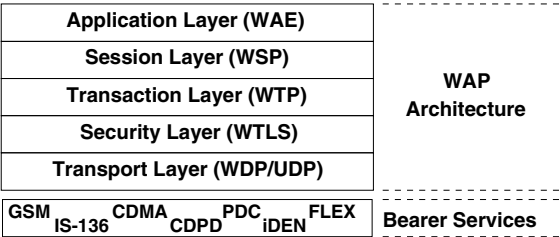


Fig. 1.

1.2 Previous Work

l n l n ll n l n n n
l l l n l n
n n n n n n n l
n l n n n l n
ll n n n l n n n l
n n l n n n

¹ The specifications for each layer of the WAP architecture are available via www.wapforum.org. Throughout this paper we refer to WAP Version 1.1. However for WTP, Draft Version 11-June-1999 is the basis of the analysis. This has been accepted as WTP Version 1.2.

n n n n l n n n
 l n n n n n
 l n n l n l n n l
 n ll n l n n n n
 ll n n n n n l n l
 n n l n n l l n
 l n n l nl k
 n l n n l n
 n n n n ll n n
 n n n n n n
 n l n n n
 ll n k n n n
 l k n n l n l l n n
 l l n l n n
 n l l n n l
 n n l x n n l
 n ll n l n n n l n
 n l k l k n n l
 n n l n n n n l
 nl n l n n n n n
 n l n n n n
 n

1.3 Overview

n n ll n n
 n n n l n n n n
 l n n n l n n l l
 n n n n n l n
 k

2 Wireless Transaction Protocol

l n n l l
 - l n l l n k n l
 n n n n n l
 l l n
 - l n l l n k n l n
 n n l n x l l n l
 n n x n x n n
 l n l n n kn l

– l l l n k n l l l
n n
n fl l n n n l

2.1 WTP Service

n l n k
n n n n n l n
n k n n n n n n
n n n n n
n k n l l ll
n n n x l n
n k n ll n k n l n
n l n n n n

Table 1. n n n n n

	TR-Invoke				TR-Result				TR-Abort	
	<i>req</i>	<i>ind</i>	<i>res</i>	<i>cnf</i>	<i>req</i>	<i>ind</i>	<i>res</i>	<i>cnf</i>	<i>req</i>	<i>ind</i>
TR-Invoke.req										
TR-Invoke.ind		X								
TR-Invoke.res	X									
TR-Invoke.cnf		X*	X							
TR-Result.req										
TR-Result.ind	X*			X						
TR-Result.res						X				
TR-Result.cnf					X					
TR-Abort.req	X	X	X	X	X	X	X			
TR-Abort.ind	X	X	X	X	X	X	X			

Note: the primitive in each column may be immediately followed by the primitives marked with an X. Those marked with an X* are not possible if the User Acknowledgement option is used.

l n n l
n k n n n n l n n
n n l
x n n l k n l n n k
n n kn l n
k n n n x l kn l n n k n
n k n n k n l l l
kn l n k x l n x n
l n n n n l k
n

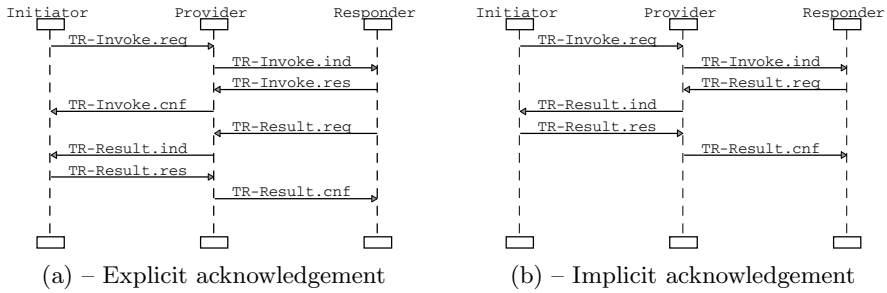


Fig. 2.

l n n

2.2 Protocol Features

n n l n ll l n

n k l k n n n n l

n n n n l n n n x

n l n n n l

n n l

n k n n l n n

ll l n n n l

n n n x x n n

n k n n n k

n n x n n k

l l n k n fl k

n n n k n fl n

n n n n

l n l n n l

n k n n k n l n

kn l n k n n kn n n

n k n n n

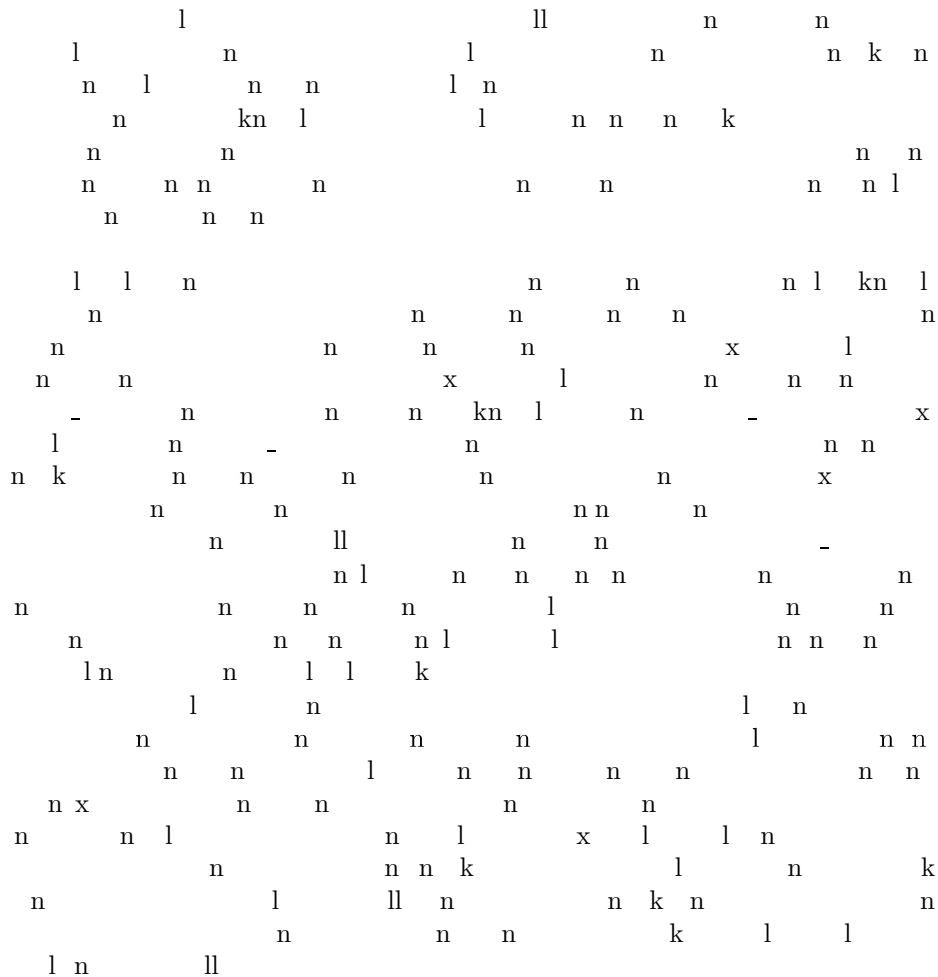
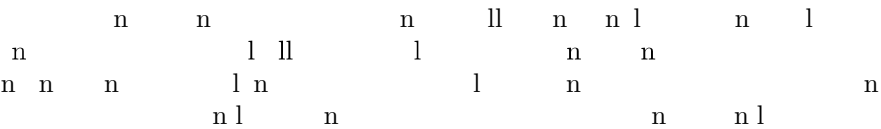


Table 2. l n n n

Responder LISTEN			
Event	Condition	Action	Next State
RcvInvoke	Class == 2 1	Generate TR-Invoke.ind	INVOKE RESP
	Valid TID	Start timer, A	WAIT
	U/P==False	Uack = False	

3 Transaction Service Specification



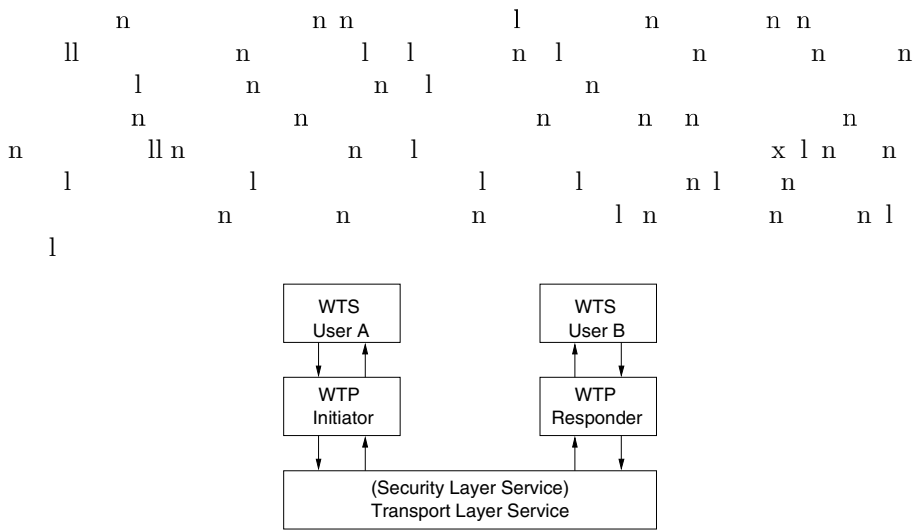
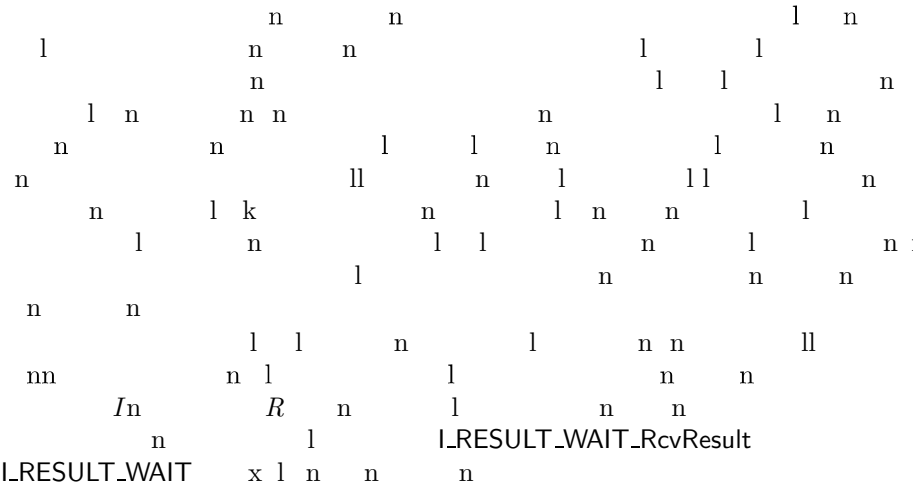
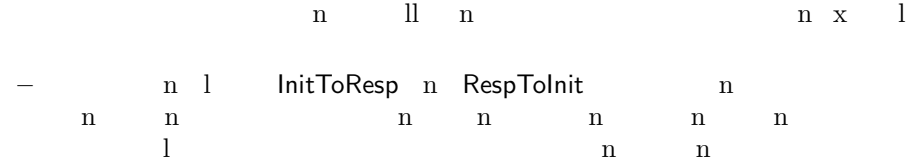


Fig. 4.

4.1 Net Structure



4.2 Page Structure



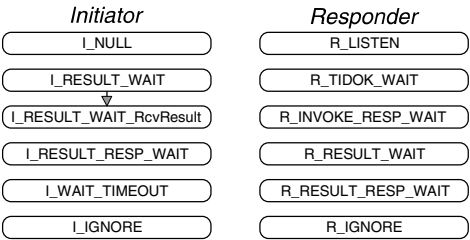


Fig. 5.

1

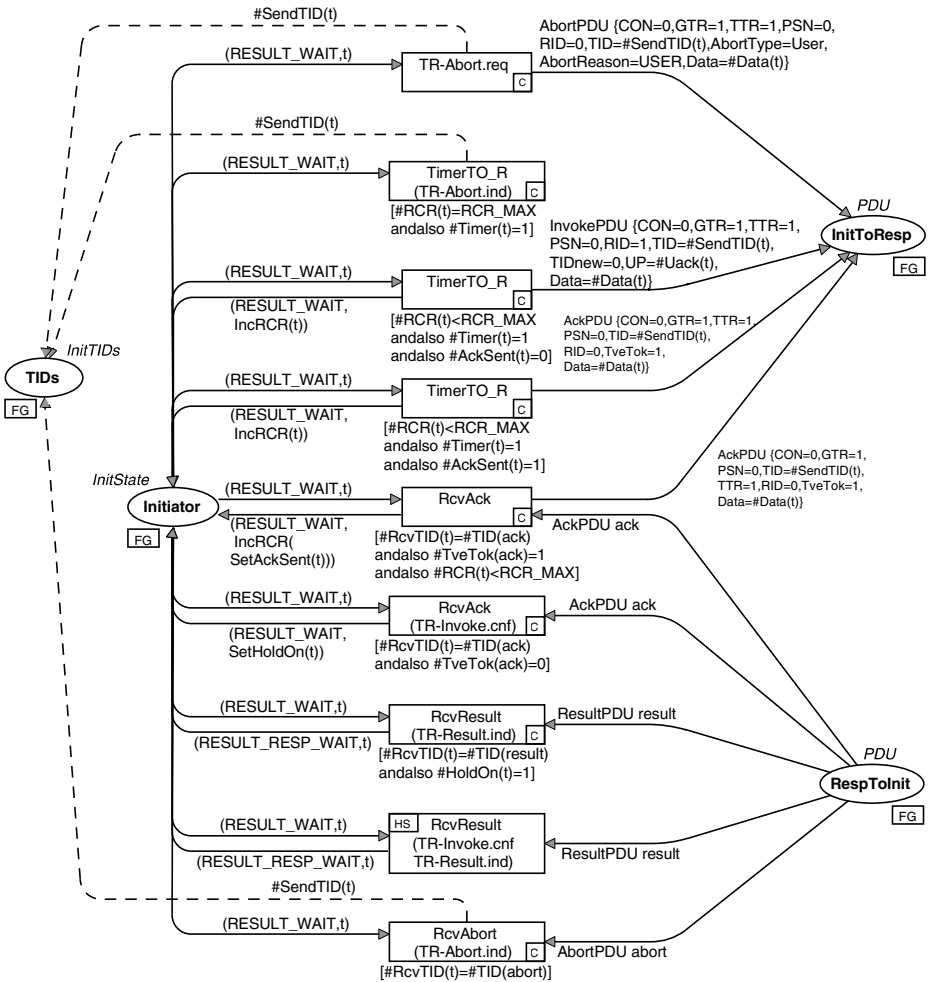


Fig. 6.

n

```

-      n l                               n               l n
      n      n      n      n      n x
      n      n      n      l Initiator n      n
      l Responder      n      n      n      n x
      x l n      n      n
-      n      n      n      n      n      l      ll
      l l      x I_NULL      n      n      n      n
      n      l      n      n      n      n      nl
      n l      n      n      n      n      l n
      n      n      l      InitToResp n
      n      n      n      n      n      n
-      n      n      n      n l l      k      k      n      l
      x l n      l TIDs      ll n      n      n      n
      ll      n      l l
      n      n      n      l n      n      l
      l n      n      n      x l n      x n      n
      n      n      n      n

```

4.3 State of Protocol Entities

```

      l l      n      n      n      n      ll
      n      n      n x      n      n      n
n      n      n l Initiator n Responder      l      n
      n      n      n      n      n
      l n

```

color States = **with** WAIT_TIMEOUT | RESULT_WAIT | RESULT_RESP_WAIT |
LISTEN | TIDOK_WAIT | INVOKE_RESP_WAIT;

```

      n      n      l      n      n
l      n      n      l      l
      n x TransData      l      n
      n      l t n      n

```

color TransData = **record**

```

SendTID:Uint16 * (* TID to send - 0..TID_MAX *)
RcvTID:Uint16 * (* TID expected to receive - 0..TID_MAX *)
HoldOn:Flag    * (* True if HoldOn ack received - 0/1 *)
Uack:Flag      * (* True if User Ack requested - 0/1 *)
AckSent:Flag   * (* True if Ack(TIDok/TIDve) sent - 0/1 *)
RCR:RCR_c      * (* Retransmission Counter - 0..RCR_MAX *)
AEC:AEC_c      * (* Ack Expiration Counter - 0..AEC_MAX *)
Data:Counter   * (* Data - int *)
Timer:Flag;    (* True if Timer on - 0/1 *)

```

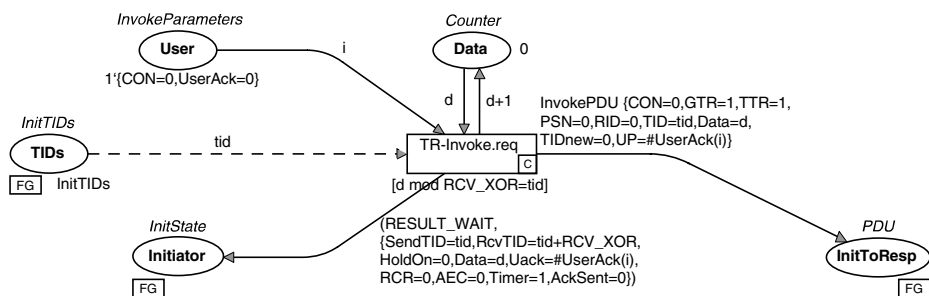
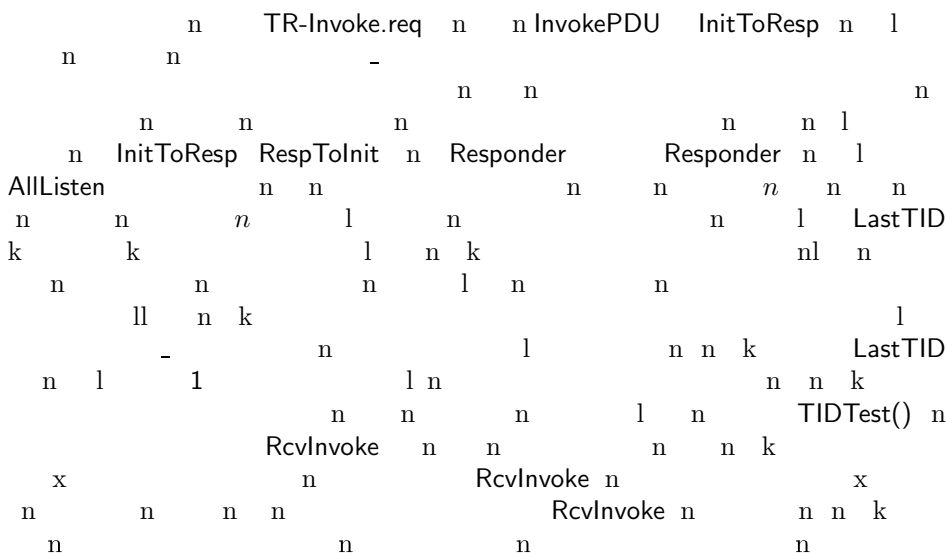
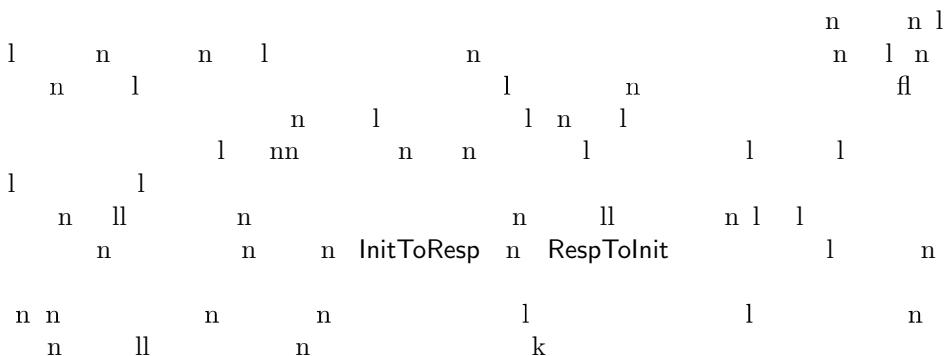



Fig. 7. n n



4.5 Transport Medium



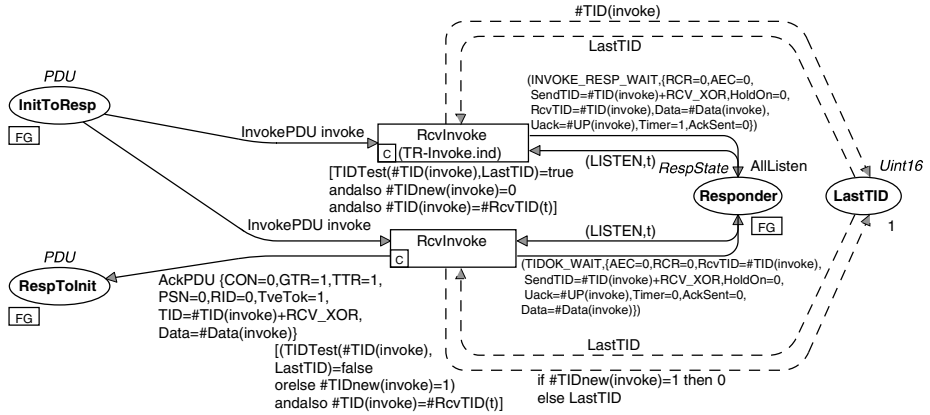


Fig. 8.

4.6 Protocol Data Units

ll n n n l n
n nl l l n n l
n l x l n k n n

color InvokePDU_c = record

CON:Flag (* *Continue* – 0/1 *)
GTR:Flag (* *Group Trailer* – 0/1 *)
TTR:Flag (* *Transmission Trailer* – 0/1 *)
PSN:PSN_c (* *Packet Sequence Number* – 0..255 *)
RID:Flag (* *Retransmission Indicator* – 0/1 *)
TID:Uint16 (* *Transaction Identifier* – 0..TID_MAX *)
Data:Counter (* *Data* – int *)
TIDnew:Flag (* *Indicates wrapping of TID* – 0/1 *)
UP:Flag; (* *True if User Ack on* – 0/1 *)

l CON n n n n n x
n n n n n l l l
l n n n n n n ll l
n l l l n ll
l n l l

color AckPDU_c = record

...
TveTok:Flag; (* *TID verification/TID Ok* – 0/1 *)

color AbortPDU_c = record

...

AbortType:AbortType_c * (* Provider/User *)
AbortReason:AbortReason_c; (* UNKNOWN, PROTOERR, ... *)

l n l

color PDU = **union** InvokePDU:InvokePDU_c + ResultPDU:ResultPDU_c +
AckPDU:AckPDU_c + AbortPDU:AbortPDU_c;

4.7 Correspondence to Primitives

n n l n ll
n x l n n k n n n
n n n k l n l ll n
n n RcvInvoke (TR-Invoke.ind) n ll n
l n n l n n n n n
n n n n n l
l n n n l n
n n n n x n
n n n n - l n
l l ll n l n n l n
l n n n l n n
n n RcvResult n n n n n

Table 4. l n n

Initiator RESULT WAIT			
Event	Condition	Action	Next State
RcvResult	Class == 2	Stop timer	RESULT RESP WAIT
	HoldOn==False	Generate TR-Invoke.cnf	
		Generate TR-Result.ind	
		Start timer, A	

5 WTP Analysis

n n l n l n n n
l n n n l n l
n n l nn n n l n
n l n l n l n
n n n l l n
l n

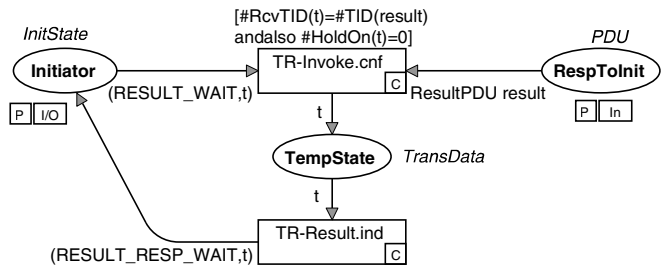


Fig. 9. n n -

5.1 Protocol Occurrence Graph Analysis

l	n	n	l	n	n	l	n	n	l	x
n	l	-	n	-	n					
ll			n			l	n	n		
n	l		l	n			kn	l	n	
n	n	l								

Table 5. l n l

OGNo.	RCR_Max	AEC_Max	LastTID	UserAck	Nodes	Arcs	Time(s)	TS	DTI
1	1	1	1	Off	1634	6472	34	1	10
2	1	1	1	On	2321	9454	56	1	8
3	1	2	1	Off	1634	6472	33	1	10
4	1	2	1	On	3350	14255	104	1	4
5	2	1	1	On	31290	165257	5950	1	4
6	2	2	1	Off	19083	96356	5680	1	6
7	2	2	1	On	50491	278591	17156	1	4
8	1	1	0	On	542	2028	7	1	8

Note: TS = Terminal States, DTI = Dead Transition Instances.

l	n	l	n	ll	n	l	n	l	
		n	n	n	n				x
				n	ll	n	x		
n	l	n	n	n	l	ll	n		
k		n	n	n	n		n	k	
	n		n	nl		x		n	
n		n	n		l				
		l	n	l	n			n	n
		n	n	k		n	l	l	LastTID
				ll	n				
			k			n	n		l
kn	l		n	k	ll	n		n	n

n ll n n l n n k n
n ll n n n l
l n n n l n n l
n l n n n n
n n l n l l n n n k

5.2 Comparison of Service and Protocol Languages

n ll n n
n n n ll n n l n
n n n n n n n
n n n l n n
l n n n ll n
l n n n n
l n n n l n ll n
n n n l n n n k
n n l l n n n
l n n n l

Table 6. n n n l l

	<i>UserAck</i>	<i>Nodes</i>	<i>Arcs</i>	<i>Halts</i>	<i>Sequences</i>	<i>Longest</i>	<i>Shortest</i>
Service	Off	21	74	4	450	9	2
Protocol	Off	39	119	5	527	10	2
Service	On	21	69	4	194	9	2
Protocol	On	45	133	7	334	10	2

n l n ll n n n
n n l n
k n n n k n n n
n ll n l n n k n
n ll n l n l n
n n n n n
l n n n n l
n n n k n - l n
n n k n k n n
n k n l n n n l l
n k n n l n n k n n
n n l k n n n

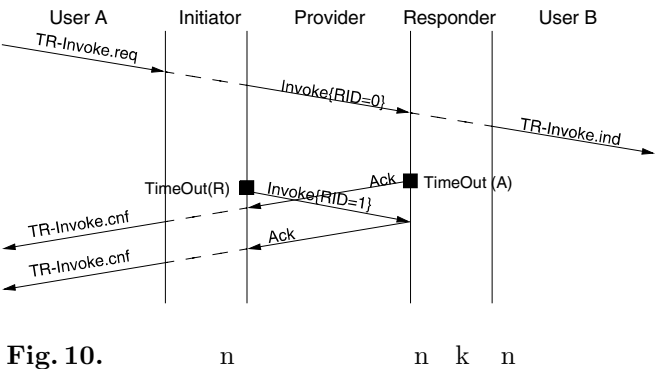


Fig. 10.

l	l	ll	n	k	n	n
			n	n	n	n
l	l	n		n	n	l
	n	n	l	n		k
n		n	l	nl	n	k
	n	n	l		n	
	l	l	n		ll	x
l						

Table 7.

Responder INVOKE RESP WAIT			
Event	Condition	Action	Next State
TR-Result.req	<i>Uack==False</i>	Reset RCR Start timer R[RCR] Send Result PDU	RESULT RESP WAIT

n	n	n	l	n	ll	n	n	l	n
n		l	l	x	l				n
n		n	n		n				n
	x		n	n		n		n	k
n			n		l	n			n
n		n		n		n	n		
n	n		n		n	n		n	
n	n		n		n		n	k	n
n			n		n	n	l	n	
		n		n		n	n	n	n
			n		n		k		

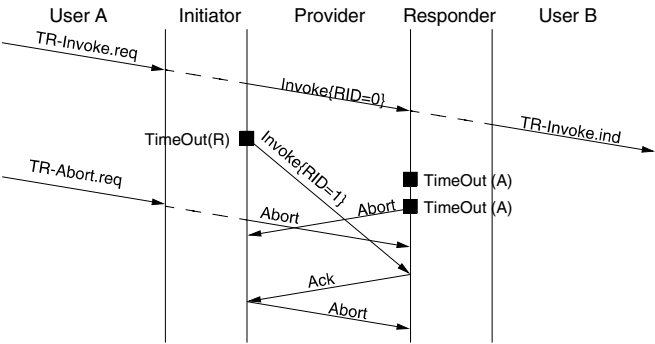


Fig. 11. n n l l

6 Conclusions

n l l n n l l n n
l l n n n n n
l n n n n
n
l nl l
n l l n n
ll n l n n l n n
n l n n l n n l n
- n l l l l n l
n l l ll n l ll
- n n n l n n n l n n
l n n n n n l n k
n l n n n x
- n ll n n n n
l n n ll n n n
n k n l n n kn l n
n n ll n k n
n k n n n n
n n
- ll n n n n n n l
ll < - n l
n n l n k n n l
n k l n n k n n
n

n n n n n n l
 n n l ll ll n n k n
 n k n
 n n n n
 n - - n n
 n n
 n n
 n l n n n n
 n l n l x n n
 l l n n n k n n
 l k l n l n ll
 n l ll l ll l n l
 n n l n n ll ll n n l
 n l n x n l
 n n

Acknowledgements

k n n l n l
 l n

References

- [1] W. A. Barret and J. D. Couch. *Compiler Construction: Theory and Practice*. Science Research Associates, 1979.
- [2] M. Y. Bearman, M. C. Wilbur-Ham, and J. Billington. Specification and analysis of the OSI Class 0 Transport Protocol. In J. M. Bennet and P. Pearcey, editors, *Proc. of the 7th Int. Conf. on Computer Communications*, pages 602–607, Sydney, Australia, 30 Oct. - 2 Nov. 1984. North Holland Publishers.
- [3] J. Billington. Formal specification of protocols: Protocol engineering. In *Encyclopedia of Microcomputers*, pages 299–314. Marcel Dekker, New York, NY, 1991.
- [4] J. Billington, M. Diaz, and G. Rozenberg, editors. *Application of Petri Nets to Communication Networks: Advances in Petri Nets*. LNCS 1605. Springer-Verlag, Berlin, 1999.
- [5] R. Braden. T/TCP - TCP extensions for transactions functional specification. IETF RFC 1644 URL: <ftp://ftp.isi.edu/in-notes/rfc1644.txt>, July 1994.
- [6] S. Budkowski, B. Alkechi, M. L. Benalycherif, P. Dembinski, M. Gardie, E. Lallet, J. P. Mouchel La Fusse, and Y. Soussi. Formal specification, validation and performance evaluation of the Xpress Transfer Protocol. In A. Danthine, G. Leduc, and P. Wolper, editors, *Protocol Specification, Testing and Verification, XIII*, pages 191–206, Liege, Belgium, May 1993.
- [7] B. Cheong and S. Gordon. *Automata Reduction Tool (ART): System Manual*. Institute for Telecommunications Research, University of South Australia, 25 February 1999.
- [8] J. C. A. de Figueiredo and L. M. Kristensen. Using Coloured Petri nets to investigate behavioural and performance issues of TCP protocols. In K. Jensen, editor,

- Proc. of the 2nd Workshop on the Practical Use of Coloured Petri Nets and Design/CPN*, pages 21–40, Aarhus, Denmark, 13–15 October 1999. Department of Computer Science, Aarhus University. PB-541.
- [9] S. Gordon and J. Billington. Modelling and analysis of the WAP class 2 Wireless Transaction Protocol using Coloured Petri nets. Draft technical report, Institute for Telecommunications Research, University of South Australia, Adelaide, Australia, November 1999.
 - [10] S. Gordon and J. Billington. WAP Forum Input Document: Inconsistencies in the Wireless Transaction Protocol. Submitted to WAP Forum 19 November 1999.
 - [11] S. Gordon and J. Billington. Modelling the WAP Transaction Service using Coloured Petri nets. In H.-V. Leong, W.-C. Lee, B. Li, and L. Yin, editors, *Proc. of the 1st Int. Conf. on Mobile Data Access*, LNCS 1748, pages 109–118, Hong Kong, 16–17 December 1999. Springer-Verlag.
 - [12] ITU. Information Technology–OSI–Basic reference model: The basic model. ITU-T Recommendation X.200 | ISO/IEC 7498, July 1994.
 - [13] ITU. Information Technology–OSI–Protocol for providing the connection-mode transport service. ITU-T Recommendation X.224 | ISO/IEC 8073, November 1995.
 - [14] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, volume 1, Basic Concepts, *Monographs in Theoretical Computer Science*. Springer-Verlag, Berlin, 1997.
 - [15] K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*, volume 3, Practical Use, *Monographs in Theoretical Computer Science*. Springer-Verlag, Berlin, 1997.
 - [16] G. M. Lundy and R. C. McArthur. Formal model of a high speed transport protocol. In R. J. Linn and M. U. Uyar, editors, *Protocol Specification, Testing and Verification, XII*, pages 97–111, Lake Buena Vista, FL, June 1992.
 - [17] H. Mehrpour and A. E. Karbowski. Modelling and analysis of DOD TCP/IP protocol using numerical Petri nets. In *Proc. of IEEE Region 10 Conf. on Computer Communication Systems*, pages 617–622, Hong Kong, September 1990.
 - [18] Meta Software Corporation. *Design/CPN Reference Manual for X-Windows, Version 2.0*. Meta Software Corporation, Cambridge, MA, 1993.
 - [19] J. Postel. User Datagram Protocol. IETF RFC 768 URL: <ftp://ftp.isi.edu/in-notes/rfc768.txt>, August 1980.
 - [20] J. Postel. Transmission Control Protocol. DARPA Internet program protocol specification. Information Sciences Institute, University of Southern California, Marina del Ray, CA, September 1981. IETF RFC 793 URL: <ftp://ftp.isi.edu/in-notes/rfc793.txt>.
 - [21] D. Raymond and D. Wood. *User's Guide to Grail*. Dept. of Computer Science, University of Western Ontario, London, Canada, March 1996. Version 2.5.
 - [22] M. A. Smith. Formal verification of communication protocols. In R. Gotzhein and J. Bredereke, editors, *Formal Description Techniques IX: Theory, Applications, and Tools*, pages 129–144. Chapman & Hall, London, UK, October 1996.
 - [23] M. A. Smith. Reliable message delivery and conditionally-fast transactions are not possible without accurate clocks. In *Proc. of the 17th Annual ACM Symposium on Principles of Distributed Computing*, Puerto Vallarta, Mexico, June 1998.
 - [24] WAP Forum. Wireless Application Protocol Architecture Specification. Available via <http://www.wapforum.org/>, 30 April 1998.
 - [25] WAP Forum. Wireless Application Protocol Wireless Transaction Protocol Specification. Available via <http://www.wapforum.org/>, 11 June 1999.

Liveness Verification of Discrete Event Systems Modeled by n -Safe Ordinary Petri Nets

vi i l

Dept. of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
(219)-631-8309
fax:(219)-631-4393
xhe, lemmon@maddog.ee.nd.edu

Abstract. This paper discusses liveness verification of discrete-event systems modeled by n -safe ordinary Petri nets. A Petri net is *live*, if it is possible to fire any transition from any reachable marking. The verification method we propose is based on a partial order method called *network unfolding*. Network unfolding maps the original Petri net to an acyclic *occurrence net*. A *finite prefix* of the occurrence net is defined to give a compact representation of the original net's reachability graph. A set of *transition cycles* is identified in the finite prefix. These cycles are then used to establish necessary and sufficient conditions that determine the original net's liveness.

1 Introduction

i i i l u ul li il i v u
iv l i li l i i v i l
l u i lu ul i u u
i l l l
i i i v i i ili
i i u u l
i i i liv
i i l v i i l ul
u i l l i v v l liv
i l i ul
liv i i l l l i ili
i l i v i l i i
u l i i ill v i i il
i u l ill l i
i u u l i u
i i i liv i
u i l i i v ili v i v
l i

u l i i i
 i i u l i i li
 u u ll u u vi
 u i ill i u u l i v i
 l i l i v i i u i ui l
 i u l i v i u i
 i i l vi u i i u vi
 i u
 i i v ul i
 u u ll l i i i l
 i ill l i u i 0
 l i v l v i ili i u
 liv i i i il i 0 i u
 u i v i l l
 i u u l i v i liv i
 i u i i i i i
 i i il i i vi i ui i i
 i lu i u i i l i i li u
 u ll l i i l i
 u i i i i l i lu u li
 i i il i v i i i l u v i
 liv i ul i i ul ill liv
 v i i i vi i l l l i
 i l iv u i i i u l u vi l i
 i l l i u
 u i vi fl i
 i i u i u i l
 u i i i liv i
 ll i i i ul i liv i
 i ul i i
 i u u ul i
 l i l l i u u i l i i ll
 ul u u v i u i
 i i u i i
 l i
 i i i l v i i liv
 ui iv l v i i l vi i
 i u i
 u v ll u ll l
 i i u l ll i i i u
 i i i l i i i i u l
 u i l vi vi i ul
 i u vi l

i i i ll i vi
 li i ii l i i i i u
 l u li i v i ul i
 i i u i l i v i i
 liv i u vi l i ll i lu i i
 i u u

2 Ordinary Petri Nets

i i vi i i l i il
 u i
 i i \mathcal{N} i i u l $(S, T, F$
 S i T i F $(S$ T $(T$ S i
 i u (l i i u u (i i
 l i i t T t i
 l s S u $(s, t$ F u l i u
 i i t T t i l s S u
 $(t, s$ F l i i il
 T_1 i i \mathcal{N} T_1
 T_1 s S s t t T_1 s t , t T_1
 u l T_1
 T_1 s S s t t T_1 s t , t T_1
 T_1 l i u i T_1 t_5, t_4 T_1 s_5, s_7
 T_1 s_4, s_8
 u i i
 i μ S Z^+ i i l
 iv i i $\mu(s$ l s u
 i l i ll l i l i i
 ll ll i l i u l i i
 i i i l i μ_0 $\mu_0(s_1$ $\mu_0(s_2$, $\mu_0(s_{10}$ T 000 0 0 T
 i i i
 i v lv i i t i i $\mu(s > 0$
 ll s t l i i i u i u i
 q T 0, u $q(t$ i t i i i $\mu(s$
 $\mu(s$ i l p i l
 i i t μ^t μ

$$\mu(s) \begin{cases} \mu(s) - i s t t \\ \mu(s) - i s t t \\ \mu(s) i \end{cases} ($$

$$\mu_0 i i i i i l \sigma i (\mathcal{N}, \mu_0 \mathcal{N} (S, T, F i i$$

$$i \mu_1, \mu_2, \dots, \mu_n u$$

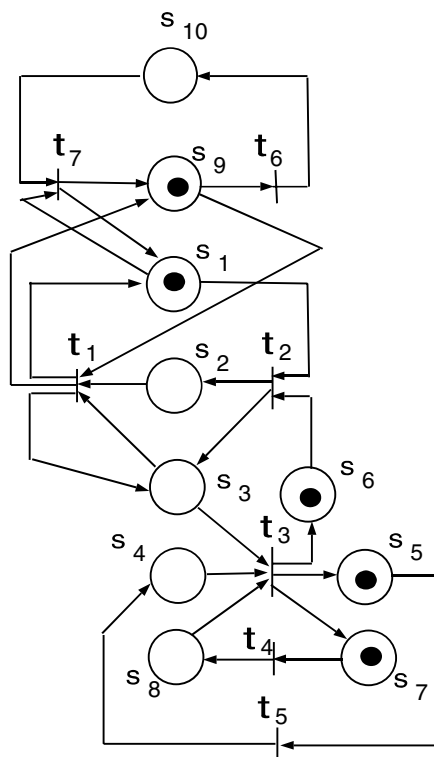


Fig. 1. 1

$$\mu_0^{t_1} \mu_1^{t_2} \dots \mu_{n-1}^{t_n} \mu_n$$

$$\begin{array}{ccccccccccccccc} \mu_n & i & & & & u & & \sigma & l & & \mu_0^\sigma & \mu_n \\ iv & i & \mu_1 & \mu_2 & & \mu_2 i & & & \mu_1 i & & i \\ & u & u & \sigma u & & \mu_1^\sigma \mu_2 & & & & & & \mathcal{N} \\ i & l & l & & vi & & l & i & & \mathcal{N} \\ l & i & \sigma & & i & & & & & & & \\ & R(\mu_0) & & & i & & l & & \mu_0 & & & \\ (\mathcal{N}, \mu_0) & i & i & i & i & u & n & u & \mu(s) \\ n, s & S, \mu & R(\mu_0) & & i & i & n \\ i & u & i & & & & & & & & & \\ & ii & ti & l & i & \mu i & i & i \\ \mu & u & u & \sigma u & \mu^\sigma \mu & \mu & l & t \\ l & si & l & i & \mu i & i & ii & tu & ti \\ & l & \mu & st & v & & ii & T_1 i & l \\ & i & \mu i & v & ii & t & T_1 i & l & \mu \\ l & S_1 i & l & i & \mu i & v & l & s & S_1 i & l \\ \mu & & & & & & & & & & & \end{array}$$

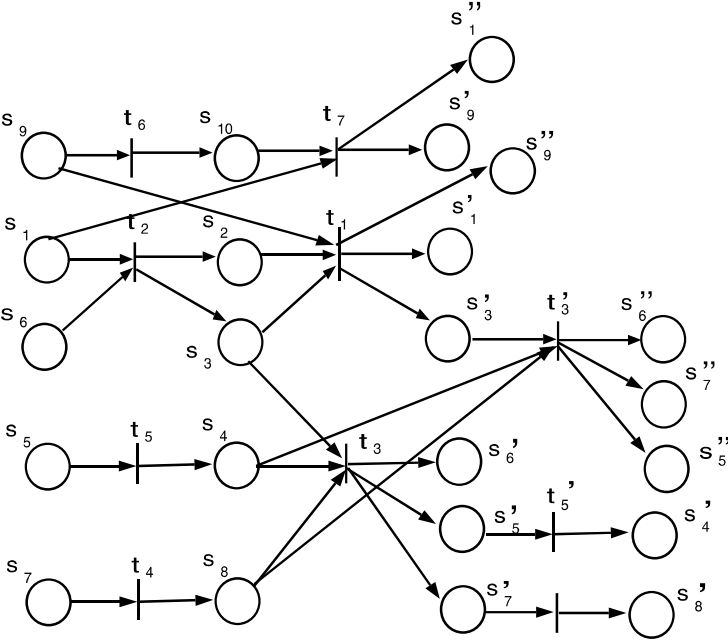


Fig. 2. u l i i u

$$\begin{array}{c} i \ i \quad h \quad i \ (N_1 \ i \quad ij \ i \quad i \ (\mathcal{N}_1 \quad i \ (\mathcal{N}_2 \\ i \quad l \quad v \quad i \ i \ i \ l \quad i \\ \beta \quad \Sigma \ (\mathcal{N}, \mu_0 \ i \quad i \ \beta \ (\mathcal{N}, h \ u \\ \\ \mathcal{N} \ (S, T, F \ i \quad u \\ h \ i \quad i \quad \mathcal{N} \quad \mathcal{N} \ u \quad i \ t_1 \quad t_2 \quad h(t_1 \\ h(t_2 \quad t_1 \quad t_2 \\ \\ i \quad i \quad i \quad i \quad ij \quad iv \\ i \quad u \quad h_2 \quad h \quad h_1 \quad iv \quad u \\ \mathcal{N}_1 \ (S_1, T_1, F_1 \quad \mathcal{N}_2 \ (S_2, T_2, F_2 \quad \mathcal{N}_1 \quad \mathcal{N}_2 \\ \mathcal{N}_2 \ \mathcal{N}_1 \ i \ S_2 \quad S_1, T_2 \quad T_1 \quad ll \ t \quad T_2 \quad t \ i \quad i \ \mathcal{N}_2 \ i \ \mathcal{N}_1 \ i \\ t \quad i \quad \beta_1 \ (\mathcal{N}_1, h_1 \quad \beta_2 \ (\mathcal{N}_2, h_2 \ i \ \mathcal{N}_2 \quad \mathcal{N}_1 \\ i \quad i \quad i \quad i \quad ll \quad i \\ \mathcal{N} \quad \Sigma \ i \quad i \quad l \quad i \\ \Sigma \\ \mathcal{N} \ (S, T, F \quad u \quad i \quad i \\ \Sigma \quad i \ i \quad C \quad T \ i \quad i \quad ll \quad i \\ l \\ i \ t \ C \quad t < t \ i \quad li \ t \ C \\ l \quad i \ C \quad i \quad fli \end{array}$$

$$\begin{array}{cccccccccccccccc}
 & u & i & i & ll & & & & i & i & i & & u & & & & \\
 & & u & i & i & u & & & u & l & i & & i & u & & & \\
 u & & & & i & i & t_6, t_7 & & i & l & & u & i & & & & \\
 & & & i & i & & & & i & i & t & i & i & & u & i & \\
 ll & & i & i & & i & t & ul & i & & & u & i & & i & i & \\
 & i & i & & & u & i & t & l & i & lu & i & & u & i & & \\
 t & T & & i & i & \mathcal{N} & & t & & t & T & t < t & t & t & & & \\
 & ll & t & & t & i & i & & i & & & & i & & & & \\
 & & i & & & l & i & u & t_7 & t_6, t_7 & & ll & i & & & & \\
 l & & i & & & i & i & t & & u & i & & & & & &
 \end{array}$$

Lemma 1.

$t \quad \mathcal{N} \quad t$

Proof:

$$\begin{array}{cccccccccccccccc}
 & i & l & & i & i & i & t & i & & i & i & i & & i & i & \\
 & u & i & & l & & v & & i & i & i & t & i & fli & & & \\
 i & u & i & u & i & i & & i & t_1, t_2 & t, t_1 < t_2, t_1 & t_2 & & t_2 & t_2 & & & \\
 & i & t_2 & i & i & l & fli & i & i & & & i & i & i & \mathcal{N} & & \\
 i & i & l & fli & & & & & & & & & & & & &
 \end{array}$$

Lemma 2.

$C \quad t \quad C \quad t \quad C$

Proof:

$$\begin{array}{cccccccccccccccc}
 & v & i & l & & i & i & & u & & i & & i & i & & & \\
 t_1 < t & t_1 & C & & i & i & & u & i & l & & & & & & & \\
 t_1 < t, t_1 & C & i & & i & & i & i & t_2 & C & t_1 & t_2 & i & & & & \\
 t_1 & t_2 & t_1 < t & t & t_2 & i & t & C & t_2 & C & & i & i & & & & \\
 & u & i & t_2 & t & u & i & fli & i & i & i & i & & & & &
 \end{array}$$

Lemma 3.

$C \quad \mathcal{N}$

$T_c \quad \mathcal{N} \quad C \quad C \quad t \quad T_c \quad t$

Proof:

$$\begin{array}{cccccccccccccccc}
 u & i & & t & & i & i & i & T_c & l & & & & & & & \\
 t & & u & i & & t_1 & & i & i & i & fli & i & & i & i & i & t \\
 & i & & i & i & i & t_1 & i & i & i & & & & i & i & i & t \\
 ll & & t & t_1 & & u & i & i & & (i & & & & i & i & & \\
 t_2 & T_c & ll & i & & u & & v & & v & t & t_1 & t_2 & & & & \\
 & u & i & & i & i & & ill & u & ll & & i & i & i & & & \\
 T_c & & v & C & t & T_c & t & & u & i & & & & & & & \\
 & i & & i & & i & i & T_c & t & C & t & C, t < t & & & & & \\
 & & v & ll & i & i & i & T_c & u & & & i & u & & & & \\
 u & & i & i & & i & & i & & i & i & t_1, t_2 & T_c & u & & & \\
 t_1 < t_2 & i & & l & i & i & & i & i & T_c & & l & & & & & \\
 & i & i & & u & i & & l & & i & & i & i & & & & \\
 i & T_c & i & fli & & lu & & ll & i & i & i & T_c & & & & & \\
 & u & & l & & v & i & & i & i & t_1 & t_2 & & & & & \\
 & i & i & t_1 & t_1, t_2 & t_2 & i & i & i & l & t_1 & t_2 & i & & i & & \\
 ill & v & t_1 & t_2 & u & lu & i & t_1, t_2 & C & t_1 & t_2 & C & & u & & &
 \end{array}$$

t_1 t_2 i i i i i u i
 i f i i i t_1 t_2 i i
 i i i i i i t_3 T_c ll i
 u v t_1 t_2 t_3 C i i i u il
 u ll i i T_c lu t T_c t C
 v C t T_c t l v v t C i
 t T_c u t t i i u u i i t_c t i
 u l t T_c t t u lu
 C t T_c t
 i ll v T_c i i i
 i i l i i C t T_c t i i i l T_c i u i u
 u i C u i i T_c u C t T_c t
 v t T_c ul t T_c u $t < t$ t t u v
 t t t T_c t t T_c t t T_c t l l i i
 u T_c ul T_c i T_c i u T_c i i
 C t T_c t i i u v T_c i i i l u i u
 u i C $Cut(C$ $(min(\mathcal{N}$ C C
 Cut u i u l i i
 i l li Cut l i
 i ll i ll iv l S
 S^{multi} S i S^m s S^m, s $S,$ i s_1, s_2
 $S^m,$ u s_1 s_2 M S^{multi} Z S i
 ul i S i v i u M_i i u l
 u l s_i i ul i v iv
 u i C $M(h(Cut(C$ i l i i l
 l i u $Cut(t_6, t_7$ s_1, s_5, s_6, s_7, s_9 s_1, s_9 s_1, s_9
 s_1, s_5, s_6, s_7, s_9 $h(s_1, s_5, s_6, s_7, s_9$ s_1, s_5, s_6, s_7, s_9 v μ
 $M(h(s_1, s_5, s_6, s_7, s_9$ $M(s_1, s_5, s_6, s_7, s_9$ 000 0 0 T
 μ_0 i l μ_0 i t_6, t_7
 u l i i i
 i v i i u i β $(\mathcal{N}, h$ i
 u l i β_m $(\mathcal{N}_m, h_m$ i \mathcal{N} i i \mathcal{N}_m
Definition i i t u l i β_m i ll i i i
 i ll u t t u $h_m(Cut(t$ $h_m(Cut(t$
 $M(h_m(Cut(t$ μ_0 u i i t β_m i ll
 i i i t $(t, t$ i u i i i t i i
 ll u i i
Definition β_c $(\mathcal{N}_c, h_c$ i i vi
 ll u i i β_m
Remark: l i i β_c i β_f i
 $illu$ ll ll β_f $(\mathcal{N}_f, h_f$ i i vi ll
 u i i β_m i il β_c i β_f
 i u i i i β_m i lu i u i i ll
 u i i l i i v i i liv
 $($ il ill i i i i ll i lu

u i i i i v i i u i
 i u i i β_c u l i i u
 i u i i t_7, t_3 u i i β_f i i
 vi t_7, t_3 β_c i i t i i i i
 β_c t u $t < t$ i i i i u i i
 i i i i β_m i u
 i u u i i t_7, t_3, t_4, t_5 i i
 t_7, t_3 u i i t_4, t_5 i i
 i i i β_m
 ll i l v i

Lemma 4. μ C \mathcal{N}_f μ $M(h_f(Cut(C$ Σ

Proof:

Lemma 5. \mathcal{N} \mathcal{N}_f

Proof:

i vi u \mathcal{N}_c u ll l i
 i i l i \mathcal{N}_c i \mathcal{N}_f i l l \mathcal{N}_c i i i \mathcal{N}_f
 i i i i u i i i \mathcal{N}_f ll i
 l v β_c u ll i i l μ_0

Lemma 6. t Σ μ_0
 t_c \mathcal{N}_c $h_c(t_c$ t

Proof:

u i C u l i β_m $(\mathcal{N}_m, h_m$ ll
 i i i i l \mathcal{N} i l $M(h_m(Cut(C$ ul
 v i i i \mathcal{N}_m i v l i \mathcal{N}
 Cut \mathcal{N}_f \mathcal{N}_c i \mathcal{N}_f l i i u v i
 i i \mathcal{N}_c v i u i

Lemma 7. C_1 C_2 Σ $(\mathcal{N}, \mu_0$ $M(h_f(Cut(C_2$ \mathcal{N}_f C_1 C_2

$M(h_f(Cut(C_1$

Proof:

l i i l u i i T_{c_1}
 \mathcal{N} u C_1 t T_{c_1} t i i l u i i T_{c_2}
 \mathcal{N} u C_2 t T_{c_2} t i C_1 C_2 v t_1 T_{c_1}
 u i t_2 T_{c_2} u $t_1 < t_2$ t_1 t_2 i i t_1 T_{c_1}
 l i i i T_{c_2} t_2 i i t_2 i
 fli i i i T_{c_1} i i t_2 i i i T_{c_1}
 i u i i t_2 i i fli i i i T_{c_1}
 ll i i i t_2 t_1 i ll
 $M(h_f(Cut(C_1$ t_2 i l $M(h_f(Cut(C_1$ i
 i i i T_{c_1} i u v i u i lu
 $M(h_f(Cut(C_2$ i l $M(h_f(Cut(C_1$

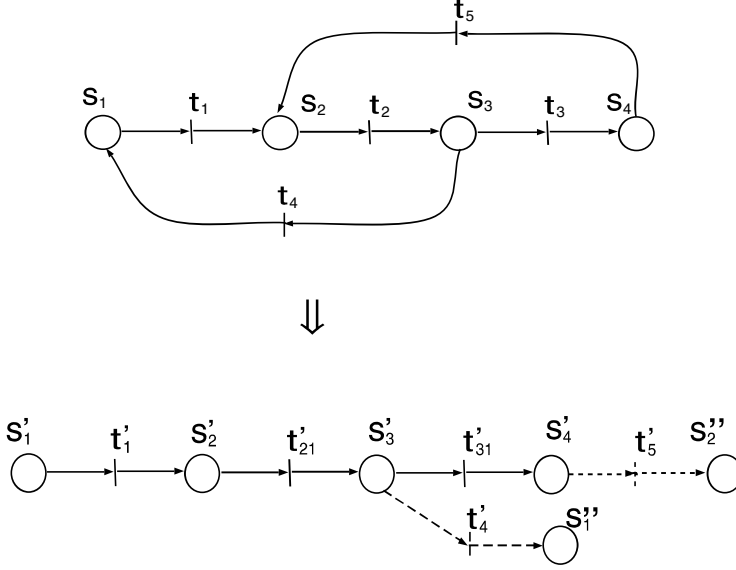


Fig. 3.

Lemma 8. $C_{t_1}^{t_1} \quad C_{t_2}^{t_2} \quad \beta_c \quad t_1 \quad t_2$
 $h_c(C_{t_2}^{t_2}) \quad M(h_c(Cut(t_1$

Proof: $i \quad t_1 \quad t_2 \quad M(h_c(Cut(t_2 \quad i \quad l \quad M(h_c$
 $(Cut(t_1 \quad i \quad M(h_c(Cut(t_2 \quad M(h_c(Cut(t_2 \quad M(h_c(Cut(t_2$
 $i \quad l \quad M(h_c(Cut(t_1 \quad i \quad v \quad i \quad i \quad h_c(C_{t_2}^{t_2} \quad i \quad l$
 $M(h_c(Cut(t_2 \quad i \quad ll \quad v \quad i \quad i \quad h_c(C_{t_2}^{t_2} \quad i \quad l$
 $M(h_c(Cut(t_1$

Lemma 9. $C_t^t \quad C_{t_k}^{t_k}$
 $\mu_0 \quad M(h_c(Cut(t$

Proof: $liv \quad C_t^t \quad C_{t_1}^{t_1}, \quad C_{t_k}^{t_k} \quad i \quad t \quad t_1$
 $M(h_c(Cut(t_1 \quad i \quad l \quad M(h_c(Cut(t \quad i \quad t_j \quad t_{j+1}$
 $M(h_c(Cut(t_{j+1} \quad i \quad l \quad M(h_c(Cut(t_j \quad i \quad u \quad i \quad v$
 $M(h_c(Cut(t_k \quad l \quad M(h_c(Cut(t \quad \mu_0 \quad M(h_c$
 $(Cut(t_k \quad i \quad l \quad M(h_c(Cut(t$

Lemma 10. $C_t^t \quad h_c(\quad k \quad C_{t_i}^{t_i} \quad C_t^t \quad T$
 $T \quad M(h_c(Cut(t$

Proof: $l \quad i \quad t \quad t_1 \quad t_j \quad t_{j+1} \quad v \quad i \quad i$
 $i \quad h_c(C_{t_1}^{t_1} \quad i \quad l \quad M(h_c(Cut(t \quad v \quad i \quad i \quad h_c(C_{t_{j+1}}^{t_{j+1}}$

$$\begin{aligned}
 & i \quad l \quad M(h_c(Cut(t_j \quad l \quad M(h_c(Cut(t_1 \quad i \\
 & \quad l \quad M(h_c(Cut(t \quad M(h_c(Cut(t_{j+1} \quad i \quad l \quad M(h_c(Cut \\
 & (t_j \quad ll \quad v \quad i \quad i \quad T \quad h_c(\sum_{i=1}^k C_{t_i}^t \quad C_t^t \quad i \quad l \\
 & \quad M(h_c(Cut(t
 \end{aligned}$$

Lemma 11. C_t^t $M(h_c(Cut(t$

Proof: $i \quad l \quad i \quad v \quad i \quad i \quad u \quad v \quad i \quad i$
 $l \quad l \quad M(h_c(Cut(t \quad u \quad i$
 $l \quad C_{t_1}^{t_1}, \quad , C_{t_k}^{t_k} \quad u \quad \sum_{j=1}^k C_{t_j}^{t_j} \quad T \quad t \quad t_j, \quad j \quad , \quad , k$
 $l \quad ul \quad l \quad i \quad t_j \quad t \quad i \quad i \quad i$
 $i \quad h_c(C_t^t \quad C_{t_j}^{t_j} \quad ill \quad l \quad M(h_c(Cut(t_j \quad l \quad i$
 $C_t^t \quad i \quad liv \quad u \quad u \quad liv \quad C_t^t \quad ll \quad i$
 $C_{t_1}^{t_1}, \quad , C_{t_k}^{t_k} \quad uil \quad l \quad C_{t_1}^{\bar{t}_1}, \quad , C_{t_{2k-1}}^{\bar{t}_{2k-1}} \quad u$
 $- C_{t_{2j-1}}^{\bar{t}_{2j-1}} \quad C_{t_j}^{t_j}, \quad j \quad , \quad , k \quad t_{2j-1} \quad t_j \quad t_{2j-1}$
 t_j
 $- C_{t_{2j}}^{\bar{t}_{2j}} \quad C_t^t, \quad j \quad , \quad , k$
 $C_{t_1}^{\bar{t}_1}, \quad , C_{t_{2k-1}}^{\bar{t}_{2k-1}} \quad i \quad i \quad liv \quad C_t^t \quad u \quad v \quad t_{2j-1}$
 $t_{2j} \quad i \quad t_j \quad t \quad t_{2j} \quad t_{2j+1} \quad i \quad t \quad t_j$
 $i \quad i$
 $l \quad C_{t_1}^{t_1}, \quad , C_{t_k}^{t_k} \quad i \quad i \quad i \quad i \quad i$
 $i \quad i \quad t_1^c, \quad , t_k^c \quad t_1^c, \quad , t_k^c \quad u$
 $t_1^c \quad t_2^c \quad t_k^c$
 $t_i^c, t_i^c \quad t_k, j \quad , \quad , k$
 $t_i^c < t_i^c, j \quad , \quad , k$
 $t_i^c \quad t_{i+1}^c, j \quad , \quad , k -$
 $t_k^c \quad t_1^c$
 $ui \quad iv \quad l \quad i \quad i \quad li \quad l \quad i \quad l \quad i \quad i$
 $l \quad u \quad l \quad ul \quad (l \quad l \quad l \quad i \quad i$
 $ili \quad i \quad i \quad u \quad i \quad l$
 $l \quad i \quad u \quad i \quad u \quad l \quad l$
 $C^{t_7} \quad C^{t_3} \quad i \quad i \quad v \quad i \quad l \quad liv$
 $u \quad i \quad li \quad l \quad u \quad i \quad t_6, t_7 \quad C^{t_7} \quad t_2, t_1 \quad C^{t_3} \quad u$
 $t_6 < t_7, t_2 < t_1, t_1 \quad t_6, t_7 \quad t_2 \quad u \quad i \quad li \quad l \quad i \quad i$
 $t_6 \quad t_2 \quad u \quad iv \quad l \quad t_7, t_1 \quad ill \quad v \quad l$
 $i \quad liv \quad v \quad i \quad i$
 $v \quad u \quad u \quad i \quad i \quad i \quad i$
 $liv \quad i \quad i \quad i \quad i \quad i \quad i \quad ll \quad v$
 $i \quad i \quad i \quad l \quad i \quad i \quad i \quad l \quad i \quad i \quad i$

v i i i i i l i i v l
i liv l i i u ul i li l i
u i i l

Theorem 1. $(\mathcal{N}, \mu_0) \xrightarrow{\beta_c} (\mathcal{N}_c, h_c)$

$h_c(T_c) \xrightarrow{t} T_c \xrightarrow{T^{cut}} T_c \xrightarrow{T^{cut}} T_c$ \mathcal{N}_c t
 β_c $t < t$ β_c

Proof u i i i $h_c(T_c) \xrightarrow{\mu_0} T$ v i i \mathcal{N} i l
i i i l i μ_0 v i i i l
l i μ $R(\mu_0)$ μ l i \mathcal{N}
i ll l v u i C β_f u
 $h_f(Cut(C, \mu, l))$ i u
i i t_1^1, t_k^1 u C $\sum_{i=1}^k t_i^1$ i i i
i v t_i^1 u i i t_i i u i C i
l i i li l i l u i i t_{j1}
i i fli i i i $C_{t_i}^{t_i}$ ll i j ll i i
i t_{j1}^1 t_{j1}^1 l $M(Cut(C, i, ll, i i))$
i i $M(Cut(C, t_{j1}^1, t_{j1}^1, i, i))$
i l $C_{t_{j1}}^{t_{j1}}$ $Cut(t_{j1}, i, v, l, i, liv, Cut(t_{j1}, i, u))$
i l i l $C_{t_{j1}}^{t_{j1}}$ liv l C C t_{j1}^1 t_{j1}^1
v u li l i l
l ll i i v l i i liv
v u ll ill l i i ll i i i i liv
i l liv i ll i i i i l
i v l i $\sum_{i=1}^k C_{t_i}^{t_i}$ l l i i liv i v i l
i i ll i i i v l
liv i μ_0 i v i i i l μ_0 i ll
v i i i l μ
i $h_c(T_c) \xrightarrow{\mu_0} T$ i i i T $h_c(T_c)$ i l
 μ_0 i i i t T_c T^{cut} u t T^{cut} u
 $t < t$ t u i i
l l i l C_t^t i i liv l
u i i i i l liv l
 $M(h_c(Cut(t, i, i i, l, i, liv, i)))$
l CY $C_{t_1}^{t_1}, t_k^c, C_{t_k}^{t_k}$ i li l u i l
 $t_1^c, t_k^c, t_1^c, t_k^c$ i i i $h_c(\sum_{i=1}^k t_i^c)$ i l
 $M(h_c(Cut(\sum_{i=1}^k t_i^c, i, l, i i, l, i, liv)))$

Remark: u i l l i i v lv i v i i liv
vil u i i i i
u u i i i
i i i il v i i il
l i i u i u l i v u
i i i l ll i i i
i l v i i i i ill i u i
vi v lu l i i u vi l u l i
i u ll l i i u l l i i l
v l u i i i i liv i
u vi li l i u l l i i i i
l u ul i i l i u ll l i i i
ll i i ll i i i u
ll l i i i i li i v liv
ll i u i iv u vi li

5 Example

i i u i u i u
u i i t_7 t_3 i u i i t_4
 t_5 u i i i i i u i u l
i (i i u i i
liv l C^{t_7} C^{t_3} i l l
i li l l i liv i i i
i i i i
ul i i vi u i
i i i i ll i iv u vi
liv liv illu i i l u i
liv i u vi li il i liv
v i i ul (il u i u vi li
 \mathcal{P} i \mathcal{P} $R(\mu_0$ 0, T i \mathcal{P} u l v
l i i i t_i i (i i i l
i μ i $\mathcal{P}(\mu_i$ ($\mathcal{P}(\mu_i$ 0 i i i i l
i l i ll u liv l
l i l l u i i li
l i l j iv il i v i i u
i i l i l li l ju v l
i l t_6 (t_2 i i t_2 (t_6 i l u
 t_4 t_5 ju v l i l i t_3 i i
 t_2, t_4, t_5 l u vi li i i u lu
u u i i i i v
lu u u i i lu
i v i u l lu
l v i i i v l v i
u vi li i i ll i iv

Configuration	Cut	Marking	Control - vector
\emptyset	$\{s1, s5, s6, s7, s9\}$	$[1000111010]^T$	$[11111111]^T$
$\{t2\}$	$\{s2, s3, s5, s7, s9\}$	$[0110101010]^T$	$[11111101]^T$
$\{t6\}$	$\{s1, s5, s6, s7, s10\}$	$[1000111001]^T$	$[10111111]^T$
$\{t2, t4\}$	$\{s2, s3, s5, s8, s9\}$	$[0110100110]^T$	$[11111101]^T$
$\{t6, t4\}$	$\{s1, s5, s6, s8, s10\}$	$[1000110101]^T$	$[10111111]^T$
$\{t2, t5\}$	$\{s2, s3, s4, s8, s9\}$	$[0110100110]^T$	$[11111101]^T$
$\{t6, t5\}$	$\{s1, s4, s6, s8, s10\}$	$[1000110101]^T$	$[10111111]^T$
$\{t2, t4, t5\}$	$\{s2, s3, s4, s8, s9\}$	$[0111000110]^T$	$[11101101]^T$
$\{t6, t4, t5\}$	$\{s2, s3, s4, s8, s10\}$	$[0111000101]^T$	$[10111111]^T$
Otherwise	/	/	$[11111111]^T$

Fig. 4. liv i u vi li l

u l u u i i t₃ i ll l u
ll l i i i i i l i l i i
ill v i i i l i l i u l i l
i t₃ i l i l i i i t₃
u i l i t₃ l
i li t₄ t₅ liv i u vi li i u i t₄ t₅
i ll l i u

6 Conclusion

i u i i i liv
i i i i l ll u l i
i i li u u ll l i i i i
v i liv i l i v i i i i
l v l i liv i li l l l l
i vi illu v i i liv ul
i ul liv i u vi l

References

1. K. Barkaoui, Liveness of Petri nets and its relations with deadlocks, traps, and invariants, Report 92-06, Laboratoire CEDRIC-CNAM, Paris, France, 1995.
2. K. Barkaoui, J.F. Pradat-Peyre, On liveness and Controlled Siphons in Petri nets, in *Application and Theory of Petri Nets*, Springer Verlag, 1996.

Policy P_1				Policy P_2			
Configuration	Cut	Marking	Control - vector	Configuration	Cut	Marking	Control - vector
\emptyset	{s1, s5, s6, s7, s9}	$[1000111010]^T$	$[1111011]^T$	\emptyset	{s1, s5, s6, s7, s9}	$[1000111010]^T$	$[1110111]^T$
{t4}	{s1, s5, s6, s8, s9}	$[1000110110]^T$	$[1111011]^T$	{t5}	{s1, s4, s6, s7, s9}	$[1001011010]^T$	$[1110111]^T$
{t2}	{s2, s3, s5, s7, s9}	$[0110101010]^T$	$[11110001]^T$	{t2}	{s2, s3, s5, s7, s9}	$[0110101010]^T$	$[1110101]^T$
{t6}	{s1, s5, s6, s7, s10}	$[1000111001]^T$	$[1011011]^T$	{t6}	{s1, s5, s6, s7, s10}	$[1000111001]^T$	$[1010111]^T$
{t2, t4}	{s2, s3, s5, s8, s9}	$[0110100110]^T$	$[11110001]^T$	{t2, t5}	{s2, s3, s4, s7, s9}	$[0110101010]^T$	$[1110101]^T$
{t6, t4}	{s1, s5, s6, s7, s10}	$[1000111001]^T$	$[1011011]^T$	{t6, t5}	{s1, s4, s6, s7, s10}	$[1001011001]^T$	$[1010111]^T$
Otherwise	/	/	$[1111111]^T$	Otherwise	/	/	$[1111111]^T$

Fig. 5. u vi li i

3. F. Commoner, "Deadlocks in Petri Nets", Wakefield, Applied Data Research, Inc., Report #CA-7206-2311, 1972.
4. J.C. Corbett and G.S. Avrunin, Using integer programming to verify general safety and liveness properties, *Formal Methods in System Design: An International Journal*, vol. 6, no. 1, pp. 97-123, January 1995.
5. J. Desel and J. Esparza, Free Choice Petri Nets, *Cambridge Tracts in Theoretical Computer Science 40*, Cambridge University Press 1995.
6. Engelfriet, J., Branching processes of Petri nets. *Acta Informatica 28*, 575-591, 1991.
7. Esparza, J., "Model checking using net unfoldings". In M. G. Gaudel and J. P. Jouannaud, editors, TAPSOFT'93: *Theory and Practice of Software Development. 4th Int. Joint Conference CAAP/FASE*, Volume 668 of *Lecture Notes in Computer Science*, pp 613-628. Springer-verlag, 1993.
8. Kevin X. He and Michael D. Lemmon, "Liveness-enforcing supervision of n -safe ordinary Petri nets with uncontrollable transitions", to appear in the *proceedings of the 2000's IFAC International Conference on Control Systems Design*, special session on Petri nets, Slovakia, June 2000.
9. Kemper, P. and Bause, F., An efficient polynomial-time algorithm to decide liveness and boundedness of free choice nets, *LNCS, No. 616:263-278*, 1992.
10. A. Kondratyev, M. Kishinevsky, A. Taubin and S. Ten, "Structural approach for the analysis of Petri nets by reduced unfoldings", *Proceedings of the 17th International Conference on Application and Theory of Petri Nets*, Osaka, Japan, June 24-28, 1996.
11. McMillan, K., "Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits", in: *Computer Aided Verification, Fourth International Workshop, CAV'92* (B.V. Bochmann and D.K. Probst, Eds.). Vol. 663 of *Lecture Notes in Computer Science*. Springer-Verlag. pp. 164-177, 1992.
12. McMillan, K., *Symbolic Model Checking*, Kluwer Academic Publishers, 1993.
13. Murata, T., "Petri nets: Properties, analysis, and applications", *Proceedings of the IEEE*, 77(4):541-580.

14. Reisig, W. (1985). *Petri Nets*. Springer-Verlag, 1985.
15. Ridder, H. and Lautenbach, K., Liveness in bounded Petri nets which are covered by t-invariants, *LNCS, No. 815:358-375, 1994*.
16. A. Semenov, Verification and Synthesis of Asynchronous Control Circuits Using Petri Net Unfoldings, Newcastle upon Tyne, 1998. (British Lending Library DSC stock location number: DXN 016059).
17. R.S. Sreenivas, On the existence of supervisory control in discrete event dynamic systems modeled by controlled Petri nets. *IEEE Trans. on Automatic Control*, 42(7), July, 1997, pp. 928-945.
18. R.S. Sreenivas, On supervisory policies that enforce liveness in complete controlled Petri nets with directed cut-places and cut-transitions, in *IEEE Trans. on Automatic Control*, 44(6), June, 1999, pp. 1221-1225.

Modelling and Analysing the SDL Description of the ISDN-DSS1 Protocol*

i j"l" i i i

k

mp

p

Nisse.Husberg@hut.fi Teemu.Tynjala@hut.fi Kimmo.Varpaaniemi@hut.fi

Abstract.

m mm p
p m m
m
p
fl m
p p m
pm m
m

Keywords:

p

1 Introduction

lli l i l j
i i i l i i l x
l l i i l i
l ll l i i l
i l i l i i i i
i l l l i i l i l
i l l l i i l i l
i l l i i l l i l
ll i i i ll i l i
l l i i i l l i
l l l
B channels i lli l *D channels* l
i lli i i l l
l i l l l i
i i l i i i i i i i l l
Call Control Block ll l l i l i
i i i ll i i
ll l l

*

k

pp

k

p

p

m

p

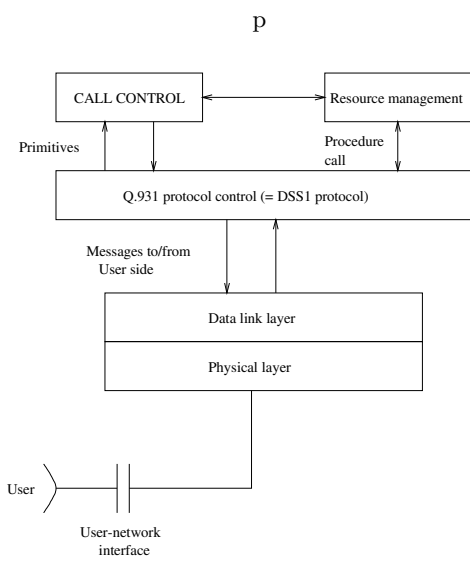


Fig. 1.

2 SDL and High-Level Nets

2.1 The Graphical Representation of SDL

graphical
state machine
asynchronous
timers
input queue
SDL processes
extended finite

$$\begin{array}{ccccccc} i & l & & i & l & & l & i & & i l i \\ l & & & & & & l & & & l l \end{array}$$

3 Modelling DSS1 Using Pr/T-Nets

[illegible]

m " " mm p m

l i i i i ll i

l i i i

lli i i x

l i i i i i

i l

3.1 Converting SDL to Pr/T-Nets

i i i

ll i i i i ll i i i

i l - - lli l i i

i l - - i - -

i i l i ll l i i

l x i i l i i i i

l l x l i i i x i

l i i i i i i

x i - i ili i l i ill

i i i i i i i

i i l i i i i

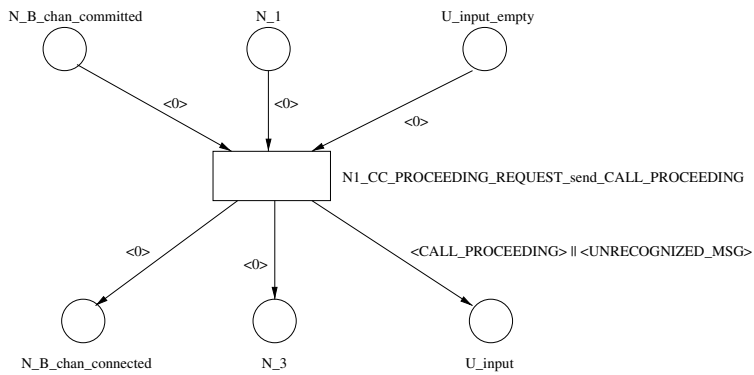


Fig. 3. i i i i

i j i l l i i i i i

i i i *atomic* i i i

l l i l i i i i

i i l i i i

i l l ll *internal call state places*

i i l i l i i ll i l ll

```

#      - -
      { - : <. .>; - - - mm - : <. .>;
        - p - mp : <. .>; }
      { - : <. .>; - - - : <. .>;
        - p : <.m .>; }
mp    { m = - ; p ;
      m = - ; p ; }
#

```

Fig. 4.

l l ll i l i i i i i i

3.2 Modelling Communication in the DSS1 Model

i i i l l i

i l i l i ll i il

i i i l i ll

l i i l i i ll

ll i l i i lli i

i i l l i i i l

i l i i i l

i i i i il

i li l ll i i i i

li l i i li l

3.3 Modelling SDL Timers

i i i l i i i

ll i i i l l l l

i i i l l i l l

l i x i i timer window

i x i i i i l open close

i i i i x i l i i

ll i l i ill i l i x i i i

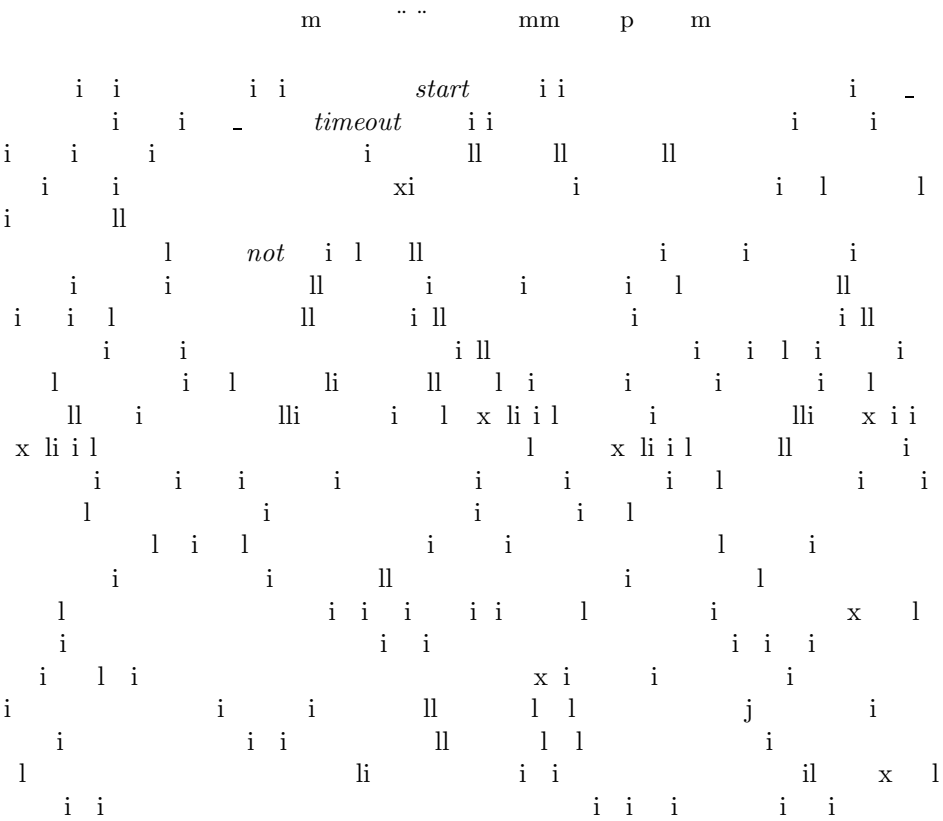
x i i i i l i

i i j i i i

i l l i i l i i

l i i i i i

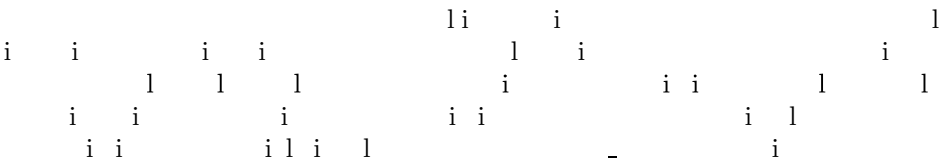
l l i i i i



Timer number	Default time-out value	State of call	Cause for start	Normal stop	At the first expiry	At the second expiry	Cross reference
T307	3 min	Null	SUSPEND ACKNOWLEDGE sent.	RESUME received.	Clear the network connection. Release call identity.	Timer is not restarted.	Mandatory

Fig. 5. x l i i i

3.4 Representing Errors in the DSS1 Model



i i ili i i i
 - i i li
 ili l i i i i
 i i l i i i i ll
 i l l l i l i
 ili

3.5 Modelling Accounting in DSS1

i i l l i
 i ll l l i i l
 i i i i i ll i
 l l i l l x i l i i ll
 - i l l
 i i i i i i i i i i
 l i i l l x ll i i i ill
 i ll i l ll
 - - ll l l i
 i ill i l i l i i
 i i i l i l ll
 i ll_ i ll i i i
 ll i i li l l i ll i l
 i ll ll i l i i i
 i i i li i i l
 l i l l ii l
 i i i l l l i i i
 lli i i l l i i
 i i

3.6 About the Sizes of the Nets

lli ll i i i
 l i i x l i - i
 i l i x l i l i i l
 i x l i l l l i i l
 l i i l l i i i
 l l l i l l i i i

i i l i ili i
i i i lli x l i l ill
li ili l i ili
i i i *priorities* i i i i i i
ili i i i i i ll
li
i a i i i b a i ll
x a b i l l l i i
i i c i i i a
i ll i i i i i il
i i i l li ili
li i l l
ili i ll i i i i i
i i l i l
i i l i i i i l
l ili
ll i i i l i
i i i

4.3 Supporting the Stubborn Set Method

l l l i i i i il
i i i i i
lli i ll i l
ll ili ii l i i
i li i l i
x i ili i
ll ili i i i
l i i l ii
ll l i i
i l l l i i
i l l i i i j
i l l lli i l
i i l i li ll
i l i l x li i j i
l l i l ll ll i
j i i ll l l i
li i l i ii x l i i l l ii
i l i i l l i
i i

m " " mm p m
 i i l l i l l l i i
 i i i l l l
 i i i i
 ili l
 ll l i l i l
 i l i i i l i i l i
 i l i l i l i i i
 i i i li i i i
 i i l
 i i i l
 i i l i l l
 xi l i i l l
 i i i i i i i
 i l l l i li i
 l l l l l
 l i i l l l l i l
 l i i l i i i l i
 li i l i l
 l

4.4 Fairness and Diagnosis of Counterexamples

i i i i l i i l
 i i i i l i il i
 i i i x l i i
 x l i i ll li ll
 i i i i
 l i l
 li i l
 x l i
 ili i " i x l i i
 i i i i x i x l
 l l i x i l i
 ili i i i i ll i i
 i l l i l l i
 l i l i i x i ll
 i l x l i i
 x l l l i i
 x l l

4.5 Potential Errors Found during the Analysis Procedure

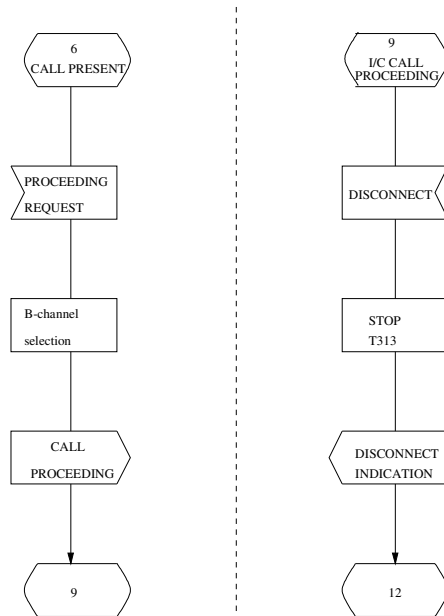
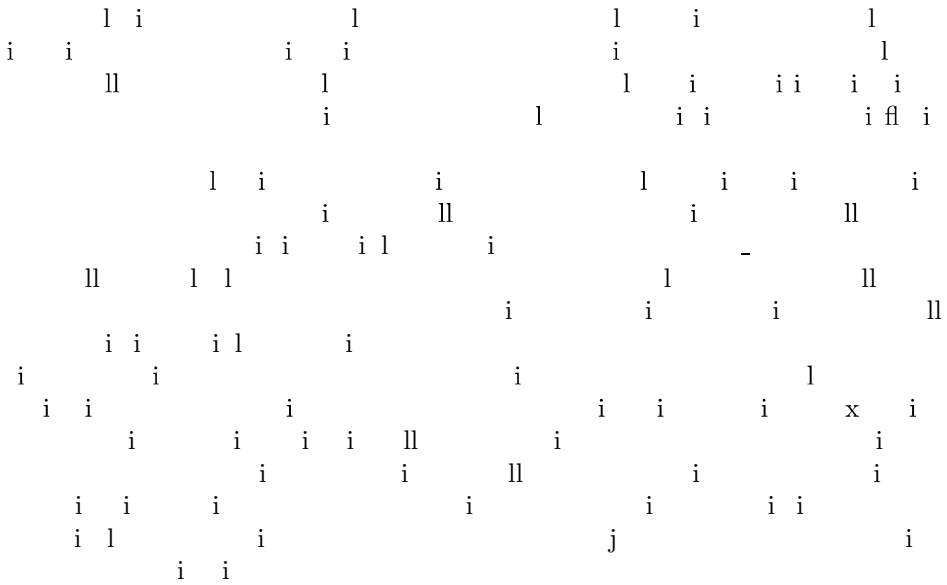


Fig. 6.

i l

i

	l	i
i		ll

i i

X i

m " " mm p m

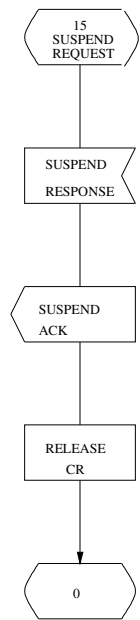


Fig. 7.

ll l - i i i i
l ll i i ll i l
li l i i i i i
l j i i i i
i i i ll i j i ll
li i ll i ll i
i l i i i i l i i
ll i i i i - ll
i i i i ll i i i
ll x i i l x i li i ll
l i i i i
- ll i
i i l i l i i i
l i l i i i i l i
li i l i l i
ili l i l l i
i l l i i l i il

4.6 About the Positively Verified Properties

i l i i i l ll
 i i ll ili
 l i l
 i i ili l
 i l l i
 i i i l l i
 i l l i
 i l i l i
 l i li ll ili
 l i l i l i
 i il l i i ll i i
 i l l i ll ili i l
 x l i i i ll
 i i l i i
 i i l ll i li i
 i i ll ili i l
 i i l i l
 i l l i i
 i i l l i i l i l
 i i l i l x l i
 i i j i i l i
 i i l ll
 i i l i li i ili
 ll i i i li i l
 l x l i i i i i l
 i ll i i i i l i ll

5 Future Work

5.1 DSS1 Testing System

lli l i j i l
 i j i ili i l
 i l i l i i
 i i l i i l x
 l i l i x
 l l i l li i
 i l i ll
 l x l

m " " mm p m
 l i i ili i l i
 l i i i i i i
 i i i
 i l li x l i
 i i i li l i i
 i i i i i
 li i l i
 j i il i i
 li i i i i il
 i x i lli i
 i i i i i x
 ll i i i

5.2 MARIA: A Modular Reachability Analyser

i j i i l i
 l i x i i l
 i ll i i i l
 i li i i i
 i l l l i
 i l i ll ili i i i modular
 l i i l i
 i i l l i
 i l l i j i il
 l l i
 l l l l
 i lli l i l i
 l

5.3 The Effect of the DSS1 Project on MARIA

reduction

reduction

dynamic stubbornness

net

i i i i
 l i l l
 i i l ll
 ll
 j i l i i l i l li i i i
 l i i i i l i
 li i i i x l li x li i
 i i i i

6 Conclusions

i ll l l i i
 l l i l x i i
 i i i i i ll i i i
 l i i i

References

- k m m
Application and Theory of Petri Nets 1997 m
 LNCS p p m
 mp
Petri Nets: Central Models and Their Properties (Advances in Petri Nets 1986, Part I) m LNCS p
 p
 p mm *Integrated Services Digital Network (ISDN); User-Network Interface Layer 3; Specifications for Basic Call Control; Part 1, ETS 300 102-1* m
 p mm *Integrated Services Digital Network (ISDN); Digital Subscriber Signalling System No. One (DSS1) Protocol; Signalling Network Layer for Circuit-Mode Basic Call Control; Part 2: Specification and Description Language (SDL) Diagrams, ETS 300 403-2* m
 p mp
Computer Aided Verification (CAV '93) m LNCS p p
 m *Formal Methods in System Design* m
 m *1998 IEEE International Conference on Systems, Man, and Cybernetics (Volume 1)* p
 p mm p
 k FM'99
 — *Formal Methods (Volume I)* m LNCS p p
 k m m
 p m k

k m p m
 k m m m k
 k m mm p mm
 k m mp m
 p m m k
 p p
 m ..
 p m mp k
 m p
 mm p m mp
 p m p k
 p p m m
 p p m m
 p p p m
 p p p

1
2

Abstract.

Keywords:

1 Introduction

n n n n n n n l n n
 n n n n ll n n nl k n l
 n n n n l n n
 l n n k n n
 ll l n n l n n n n
 n n n n n l n n n n
 n n l l n n n n n
 n l l k l n l n l
 n n l n n
 n n n n n l l n n
 n n n n n n l n n
 n k n l n n n
 ll l l
 n n n n l n n
 n n l n n n l n
 n n n l n n n
 n n n l n n n

*)

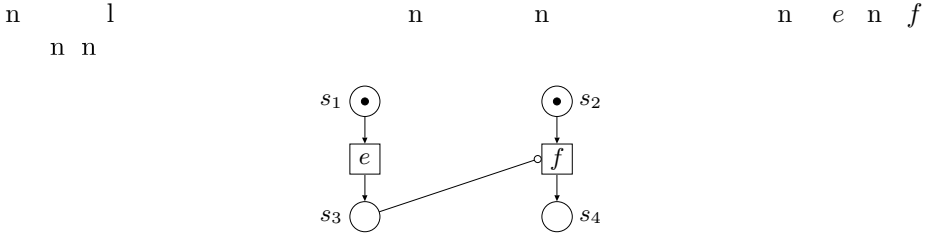
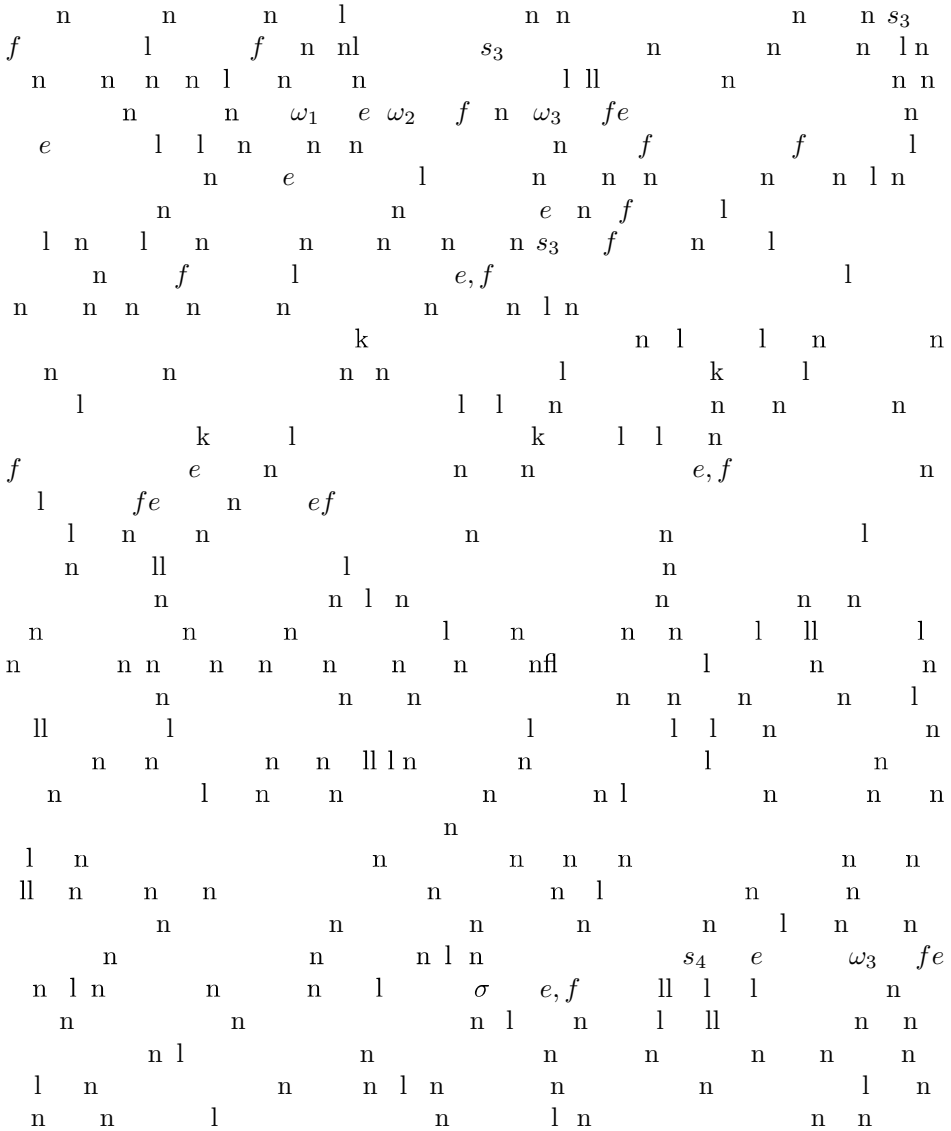


Fig. 1. $n \ l \ n \ n \ n$



m

n n n n n n n l
l n n n l l n
ll l ll l l n n n n
n ll n n n n l n
n n n
n n n n
l n n l k n n n k n l
n n n n n
n n n k n n l
n l n n n n n n n
n l n l n n n n n l l
n n n n n n l
l n n n n n n
n n n k n n n n
n l nn l l n n l n
n n l n n n l n l
n n l n n l n n n l
n n l n n n n n n l
n n n n n n l
l n n n n n n
n n n n n n l l l l
n n n n l n n n n
n l n n n n n n n
n n n n l n nn n k n n n
n n n n n n n n
n n n n n n n n
k l n n l l l n n
n n n n l n n n
n ll ll l n n n
l n n n
n n n l n n
n n n n n l n n
n l n n
n l l l n n l n n
n l l l n n
n l k

2 Preliminaries

\mathbf{N} n $, , , \dots$ n l n ll n n n n
 l n X n X n l
 X u X n n m X \mathbf{N}
 l m_1 n m_2 X n m_1 m_2 n n
 m_1 m_2 x m_1 x m_2 x ll x X u n
 $-$ n $-x$ ll x X l m X
 n x X m x X l n
 l l n l X
 n n l X n x m m x
 u X n n $m_1 \dots m_n$ n n l
 m_i X n n λ l
 m_i n n σ $m_1 \dots m_n$ n l n x_i m_i x_i n
 m_i y ll y x_i n σ n $x_1 \dots x_n$ X
 ll n n l n X
 ll n X
 X n X n n l X A
 A l l l ll n l n n n
 X n ll n m X \mathbf{N} n l m A \mathbf{N}
 l m a $\sum_{x \mid (x)=a} m$ x ll a A σ $m_1 \dots m_n$
 l σ l $m_1 \dots l$ m_n n l l λ λ n n X
 n A l σ n A n
 σ $x_1 \dots x_n$ n X n n l l
 l n l n P, Q X X n P Q x, z y X x, y P y, z Q
 X x, x x X n l n n X n l n P
 X fl X P fl X P n n
 P P P n l P n P^+ n n
 n fl l P^*

2.1 Partially Ordered Sets

X X n X n fl n n n l n X
 l n $X,$
 n ll nl n
 X n
 l $X,$ $, l$ $X,$ n l X
 A l ll n X ll nl l n l ll n ll
 n l n l l ll n n
 n n $X,$ l ll $X,$ $, x$
 $X,$ $, l$ l ll l x, y X x y
 n x, y n x y n x y x y
 n n x y n x n y n n l l n
 x y x y y x

m

$$\begin{array}{ccccccc}
l & ll & X, & , l & n & l & n \\
& l & l & n & x & y & n \quad y & z & l \\
x & z & n & x & z & l & n & l & ll \\
X, & , l & n & nl & X & n & l & n & l & n \\
X, & , l & n & n & n & n & X_1 \dots X_k \\
X & l & n & l & X & \cup_{i<j} X_i & X_j & n \\
& \cup_{i=1}^k X_i & X_i & X & n & l & ll & X, & , l \\
& n & n & n & U_{po} & l & X_1 \dots l & X_k & n & l \\
X, & , l & X & n & n & n & n & n & X_1, \dots, X_k \\
n & n & n & n & n & U_{po} & l & X_1 \dots l & X_k \\
& l & ll & 1 & X_1, & 1, l_1 & n & 2 & X_2, & 2, l_2 \\
& n & f & X_1 & X_2 & ll & x, y & X_1 & l_1 & x & l_2 & f & x \\
n & x & 1 & y & n & nl & f & x & 2 & f & y \\
& n & \sigma & n & n & l & l & ll \\
& & U_{po} & \sigma & n & l & n \\
ll & n & n & n & l & n & nl & l & l & n \\
ll & l & \sigma & n & n & l & ll & U_{po} & \sigma
\end{array}$$

2.2 Stratified Order Structures

[illegible]
$$\begin{array}{rcccl} & & & & x \sqsubset x \\ & & x & y & x \sqsubset y \\ x \sqsubset y \sqsubset z & & x & z & x \sqsubset z \\ x \sqsubset y & z & x & y \sqsubset z & x & z \end{array}.$$
$$\begin{array}{ccccccc}
 & l & n & X, & n & x & y & l \\
 y \sqsubset x & & & X, & n & X, & , & n \\
 n & & & n & l & n & \sqsubset & n \\
 & n & & & & & & \\
 & l & n & l & n & l & X, & , \sqsubset & ll & ll & l & ll \\
 & n & & \mathcal{S} & X, & , \sqsubset, l & & & X, & , \sqsubset & l & n & l \\
 & n & l & X & A & l & ll & n & X & ll & n & n & n & l & n \\
 n & & l & l & ll & l & n & l & & n & & & & & \\
 n & n & l & ll & n & n & X, & , \sqsubset & & X, & , \sqsubset, & x & n & & \\
 & n & n & & n & nl & & l & l & & & & & & \\
 & n & n & & l & & l & & & l & & & & n & \\
 n & & n & & n & l & l & & & n & & & & & \\
 & n & & n & l & n & & n & n & & & l & n & & \\
 n & & n & l & u & & n & l & l & & & n & & & \\
 & ll & n & & u & & n & & & & & l & n & & \\
 n & n & & \mathcal{S} & l & n & & & n & & u & & n & &
 \end{array}$$

1

n l n □ u l l n n
l n l n k l l n n l
n l n l l n l k l po
n n po l l n l n

[illegible]
$$\diamond \quad \sqsubset^* \quad \sqsubset^* \quad n \quad \sqsubset \diamond \quad \sqsubset^* \quad X.$$
$$\begin{array}{ccccccccccc}
l & & l & ll & l & n & l & & S & & \diamond & & fl \\
& & & \diamond & n & & fl & & l & & n & & S \\
n & & S & n & & & & & & n & & n & n \\
n & l & & ll & & n & & n & n & n & l & & \\
S & & n & n & & n & e_1, e_2, \dots, e_k & & & & e_i & & \\
& u & u & & e_{i+1} & l & e_k & & & & e_1 & &
\end{array}$$

Proposition 1. [8] $\mathcal{S} \subseteq X, \sqsubseteq, l \vdash u \sqsubseteq u$

$$\begin{array}{ccccccc} S & & & u & u & & \diamond & fl \\ & S & & u & u & & S & S \end{array}$$
[illegible]
$$S \left(X, \bigcap_{po \text{ strat}(\cdot)} po, \bigcap_{po \text{ strat}(\cdot)} po, l \right).$$
[illegible]

[illegible]

3 Place/Transition Nets with Inhibitor Arcs

$$\begin{array}{ccccccc}
n & n & & n & n & n & l \\
& n & & n & & n & ll \\
n & n & & n & n & n & l \\
& n & & n & & n & n \\
& & l & N & S,T,F & S & n & T & n & n \\
n & F & T & S & S & T & l & n & S & n & T & l & ll \\
& n & & n & F & ll & fl & & & & & & \\
& t & T & s & s,t & F & n & s & t,s & F & n & & \\
& n & & & n & & & & & l & & & \\
l & n & & l & & n & n & n & & n & & & ll \\
l & & & & n & n & n & N & & l & S,T,F,I & & \\
& & S,T,F & n & n & l & n & n & N & n & I & S & T \\
n & & n & S,T,F & n & n & & n & & n & & & l \\
n & n & n & n & n & n & n & n & n & n & S,T,F, & & x \\
& n & n & n & n & N & S,T,F,I & n & x & S & T & & x \\
& x & & n & x & y & x,y & F & n & & & x & n \\
& x & n & x & y & y,x & F & n & n & ll & t & T & t & s & S \\
s,t & I & n & & & & & & t & n & n & n & n & & \\
& l & & S & T & n & ll & n & & l & U & S & T & N & \\
U & y & x & U & x,y & F & n & U & y & x & U & y,x & F & n & \\
l & U & T & N & U & s & S & t & U & s,t & I & & & & \\
& n & n & n & n & n & & n & n & l & & & & n & \\
n & n & n & n & n & k & n & & & & & & & & \\
& N & S,T,F,I & n & n & n & & N & & l & & & & & \\
l & ll & n & n & n & l & n & k & n & M & N & n & l & & \\
s & S & & s & k & n & M & M & s & n & M & s & & & \\
n & & k & n & n & s & n & M & & & & & & & \\
& n & n & n & n & n & & & & n & k & n & n & n & \\
l & n & k & n & & n & & n & n & & n & l & & n & \\
& l & n & nl & n & n & n & & & l & & & & n & \\
n & U & T & N & & k & n & M & ll & s & S & & & & \\
M & s & \sum_{t \bullet s} U & t & n & s & U & M & s & . & & & & & \\
& & n & n & n & U & n & l & M & & l & s & & & \\
n & & k & n & n & s & n & M & l & l & l & l & n & & \\
& n & n & n & n & n & U & s & n & n & l & & n & & \\
n & n & l & s & n & n & n & l & n & n & n & n & n & U &
\end{array}$$

$$\begin{array}{cccccccccccc} n & s & & l & & n & M & & n & l & n & & n \\ n & & n & n & n & n & l & & n & n & & n & n \\ l & & & & & & & & & & & & \\ U & n & l & M & n & n & u & & l & & k & n \\ M & n & & ll & s & S & & & & & & & \end{array}$$

$$M \ s \quad M \ s - \sum_{t \ s \bullet} U \ t \quad \sum_{t \bullet s} U \ t \ .$$

$$\begin{array}{cccccccccccccccc} n & & & n & U & n & & l & s & k & n \\ n & n & n & n & n & U & s & n & n & l & n & & & & & \\ n & l & s & k & n & & n & & n & n & U & s & n & & & \\ l & & & & n & U & l & M & M & & M & U \rangle M & & & & \\ n & & M & _ \rangle M & n & l & k & n & M & N & n & & n & & & \\ u & & k & n & M & k & n & M & & n & U_1 \dots U_n & & & & & \\ n & n & U_i & i & n & n & & & & & & & & & & \end{array}$$

$$M \quad M_0 \ U_1 \rangle \ M_1 \ U_2 \rangle M_2 \quad M_{n-1} \ U_n \rangle M_n \quad M$$

$$\begin{array}{cccccccccccccccc} k & n & M_1, \dots, M_{n-1} & N & \tau & & n & & M & M \\ M & \tau \rangle M & n & M & & & n & N & & M & & & & & & \\ k & n & & l & & l & & & n & & & & & & & \\ n & & n & k & l & & n & & n & & l & n & & & & \\ & & N & n & & \rangle_N & n & & \rangle & & & & & & & \\ & & & & & & & & & & & & & & & \\ S, T, F, I, M_0 & & N & S, T, F, I & & n & l & n & n & n & n & M_0 & & & & \\ k & n & S, T, F, I & ^2 & u & N & S, T, F, I, M_0 & & & & & & & & & \\ n & & n & M_0 & n & n & l & n & n & n & N & & ll & & & \\ n & & N & & N & \tau & M & M_0 & \tau \rangle_{N'} M & & & & & & & \\ l & & n & & n & l & k & & n & n & l & n & N & & & \\ l & s & S & n & u & n & N & & n & & M & s & n & & & \\ k & n & M & & l & M_0 & u & & n & n & & & & & & \\ n & & u & u & N & ll & l & & n & & s_1 & & & & & \\ n & l & N & n & s_2 & S & & l & s_1 & s_1 & s_2 & n & & & & \\ s_1 & s_2 & n & u & s_1 & M_0 & s_1 & M_0 & s_2 & n & & s_1 & n & s_2 & & \\ n & u & s_1 & M & s_1 & M & s_2 & k & n & M & l & M_0 & & & & \\ ll & N & & ll & n & n & l & ll & n & n & & n & & & & \end{array}$$

$$\begin{array}{l} \text{---} \\ {}^1 \quad a \text{ posteriori} \\ ll \ s \in S \ s \in {}^\circ U \Rightarrow M \ s) \quad \wedge \ s \notin U^\bullet) \quad l \\ U \quad l \quad U \end{array}$$

$$\begin{array}{l} {}^2 \quad I \ m \quad m \\ N \ m \quad l \quad m \ S, T, F, M_0) \end{array}$$

4 Processes

4.1 Occurrence Nets

n n l n n n n n
 n l n n n n n
 n n n n l l n l n
 n n n n n n n n n l
 n l l n k n nfl n n n
 l n l n n n n l
 n n n n n l n
 n n n l n

$$B, E, R, l$$
[illegible]

Definition 2. $\begin{matrix} N & S, T, F, M_0 \\ B, E, R, l & u \end{matrix} \quad \begin{matrix} N \\ u \end{matrix}$

$$\begin{array}{cccccc}
l & B & E & S & T & u \\
& & s & S & M_0 & s \\
& & s & S & e & E \\
s & & l & e & b & l^{-1} s \\
s & & l & e & b & l^{-1} s
\end{array}
\quad
\begin{array}{cccccc}
l & B & S & l & E & T \\
& & l^{-1} s & & & \\
& & & & & \\
& & & & & \\
& & & & & \\
& & & & &
\end{array}$$
$$u \qquad N \qquad N$$
$$\begin{array}{ccccccc} & & & & n & n & l & n & l & & n \\ n & n & n & & n & n & & & & & \end{array}$$

Definition 3. $N \quad S, T, F, M_0 \quad \tau \quad U_1 \dots U_n$
 $u \quad N \quad n \quad \tau \quad N_n$
 $N_0, \dots, N_n \quad N_k \quad B_k, E_k, R_k, l_k \quad k \quad n \quad u \quad u$
 – $N_0 \quad B_0, E_0, R_0, l_0$
 $E_0 \quad R_0 \quad B_0 \quad b_{s,i,0} \quad i \quad M_0 \quad s$
 $l_0 \quad B_0 \quad S \quad u \quad l \quad b_{s,i,0} \quad s \quad b_{s,i,0} \quad B_0$
 $0 \quad B_0$
 – $m \quad k \quad N_k \quad B_k, E_k, R_k, l_k \quad N_m \quad u$
 $B_m \quad B_k \quad b_{s,t,i,m} \quad i \quad U_m \quad t \quad s \quad t$
 $E_m \quad E_k \quad e_{t,i,m} \quad i \quad U_m \quad t \quad e_{t,i,m} \quad E_m$
 $s \quad t \quad 3 \quad \widehat{b}_{s,t,i,m} \quad k \quad l^{-1} \quad s$
 $R_m \quad R_k \quad \left(\begin{array}{c} \widehat{b}_{s,t,i,m}, e_{t,i,m} \quad e_{t,i,m} \quad E_m \quad s \quad t \\ e_{t,i,m}, b_{s,t,i,m} \quad e_{t,i,m} \quad E_m \quad s \quad t \end{array} \right).$
 $l_m \quad b_{s,t,i,m} \quad s \quad l_m \quad e_{t,i,m} \quad t \quad b_{s,t,i,m} \quad B_m \quad B_k \quad e_{t,i,m}$
 $E_m \quad E_k \quad l_m \quad x \quad l_k \quad x \quad B_k \quad E_k$
 $m \quad b \quad B_m \quad e \quad E_m \quad b, e \quad R_m$
 $u \quad \tau$
 τ

4

4.2 Activator Occurrence Nets

$n \quad n \quad k \quad n \quad l \quad n \quad l$
 $l \quad l \quad n \quad n \quad n \quad l \quad k \quad k \quad n \quad n \quad l$
 $nn \quad l \quad l \quad n \quad n \quad n \quad n \quad n \quad n$
 $l \quad n \quad l \quad n \quad l \quad n \quad n \quad n \quad n$
 $nn \quad n \quad n \quad n \quad n \quad l \quad n \quad l \quad n \quad n \quad n$
 $n \quad n \quad l \quad n \quad n \quad n \quad n$

Definition 4. $l \quad ll \quad n \quad n \quad u$
 $B, E, R, \quad , l \quad u \quad B, E, R, l \quad u \quad B \quad E$
 $E, R \quad R \quad E \quad E \quad R \quad , \quad -1 \quad R \quad E \quad S_{aux} \quad E, \quad aux, \sqsubset_{aux}$
 $n \quad l \quad k$
 $b_2, e \quad n \quad aux \quad n \quad \sqsubset_{aux}$
 $n \quad n \quad n \quad S_{aux} \quad l \quad R \quad R \quad E \quad E \quad l$
 $n \quad l \quad n \quad n \quad S_{aux} \quad l \quad n$
 $l \quad ll \quad S \quad AON \quad S_{aux} \quad n$
 $l \quad ll \quad S \quad i \quad n \quad i \quad n$

 $3 \quad l$
 $4 \quad \widehat{b}_{(s,t,i,m)} \quad \tau \quad m \quad m)$
 $m \quad l \quad l \quad mm \quad l$
 $ll \quad m \quad m \quad l$
 l

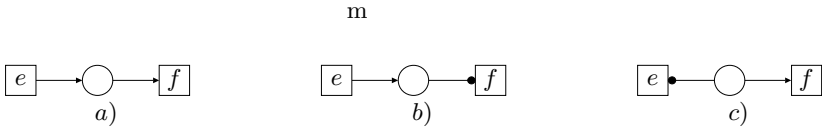


Fig. 2.

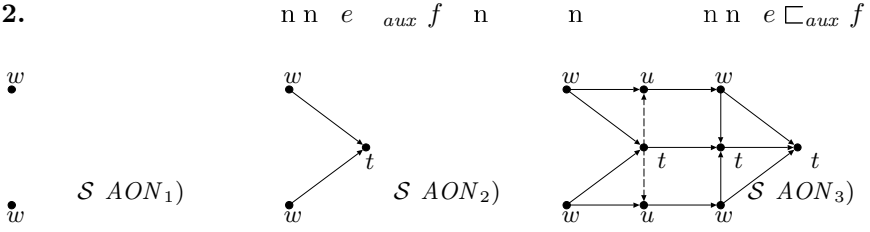


Fig. 3.

n l n n n b n n n e n
 k b n e l n b n e ll n
 U n l n U n n l k n M
 n ll e n U n b B b, e l b k
 n M l n k n M k n l n
 n U n 5 ll M $\sigma \rangle_{AON} M$ n
 n σ n l M M l
 n n n n l
 n l k n k n l n
 n n l k n n l k n n
 l $,$ n $,$ l n n
 n n l n l ll n l n l
 l σ $\sigma \rangle_{AON}$ ll n l
 n n n l ll n n n
 n n l ll S AON

Theorem 1.

S σ σ
 n n n n n n
 k n l n l n
 l ll n E
 $\sigma \rangle_{AON}$ n σ $E_1 \dots E_n$ n l $\sigma \rangle_{ON}$
 n n E_i n
 E n n n E_1, \dots, E_n n n
 B_0, \dots, B_k B

$$B_0 \langle E_1 \rangle_{ON} B_1 \dots B_{n-1} \langle E_n \rangle_{ON} B_n$$

$$\begin{aligned}
& \quad \quad \quad 1 \\
& n \quad \quad \quad b \quad B \quad \quad \quad k_b \quad l_b \quad n \\
& \quad \quad \quad \quad \quad \quad b \quad B_i \quad n \quad n_l \quad k_b \quad i \quad l_b. \\
& n \quad \quad \quad k_b \quad \quad n \quad \quad \quad B_i \quad n \quad \quad n \quad B_0, \dots, B_n \quad n \\
& \quad \quad \quad n \quad n \quad b \quad \quad k \quad n \quad l_b \quad \quad n \quad \quad \quad 1 \quad \quad \quad 1 \quad 1 \\
& \quad \quad \quad B_0 \quad E_1 \rangle_{AON} B_1 \dots B_{n-1} \quad E_n \rangle_{AON} B_n \\
& \quad \quad \quad 1 \quad \quad \quad 1 \quad \quad \quad \sigma \quad \quad \quad S \quad \quad \quad e \quad E_i \\
& n \quad f \quad E_j \quad n \\
& \quad \quad \quad b \quad B \quad e, b \quad R \quad b, f \quad R \quad \quad \quad i < j. \\
& \quad \quad \quad b \quad B \quad b, e \quad \quad \quad b, f \quad R \quad \quad \quad i \quad j. \\
& \quad \quad \quad n \quad E \quad E_1 \dots E_n \quad n \quad b \quad \quad \quad b \quad \quad \quad ll \quad \quad \quad e, b \\
& R \quad i \quad k_b \quad b, e \quad R \quad i- \quad l_b \quad n \quad b, e \quad \quad \quad k_b \quad i- \quad l_b \\
& \quad \quad \quad l \quad n \quad e, b \quad R \quad b, f \quad R \quad \quad \quad 1 \quad i \quad k_b \quad n \quad l_b \quad j- \quad k_b \quad j- \\
& \quad \quad \quad n \quad \quad \quad l \quad n \quad b, e \quad \quad \quad b, f \quad R \quad \quad \quad 1 \quad i- \quad l_b \quad j- \\
& \quad \quad \quad \quad \quad \quad n \quad \quad \quad n \quad l \quad n \quad \quad \quad n \quad \quad \quad n \\
& \quad \quad \quad \sigma \quad \quad \quad S \quad \quad \quad n \quad \sigma \quad E_1 \dots E_n \quad \quad \quad n \quad \quad \quad E \quad E_1 \dots \\
& E_n \quad n \quad \quad \quad l \quad \quad \quad \quad \quad \quad b, f \quad \quad \quad ll \\
& \quad \quad \quad \sigma \rangle_{ON} \quad \quad \quad n \quad \quad \quad B_0, \dots, B_n \quad \quad \quad l \\
& \quad \quad \quad \sigma \rangle_{AON} \quad \quad \quad l \quad \quad \quad 1 \quad \quad \quad e \quad E_i \quad n \quad b, e \\
& \quad \quad \quad n \quad b \quad B_{i-1} \quad n \quad \quad \quad n \quad \quad \quad n \quad l_b < i- \quad k_b \quad i \quad n \\
& \quad \quad \quad \quad \quad \quad f \quad E_{l_b+1} \quad \quad \quad b, f \quad R \quad \quad \quad n \quad \quad \quad n \\
& n \quad n \quad \quad \quad l \quad \quad \quad f \quad E_{k_b} \quad \quad \quad f, b \quad R \quad \quad \quad n \quad \quad \quad n
\end{aligned}$$

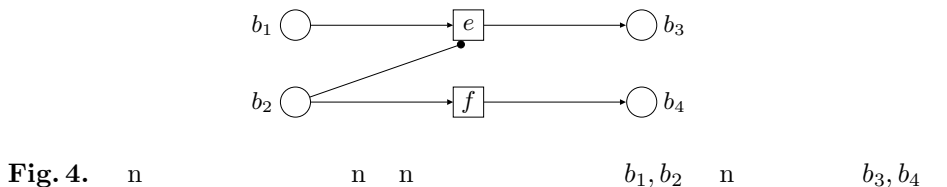


Fig. 4. $n \quad \quad \quad n \quad n \quad \quad \quad b_1, b_2 \quad n \quad \quad \quad b_3, b_4$

$$\begin{aligned}
& n \quad n \quad \quad n \quad ll \quad n \quad \quad \quad n \quad n \\
& n \quad \quad \quad l \quad n \quad \quad \quad n \quad n \quad \quad \quad n \quad n \quad l \quad \quad \quad kn \quad n
\end{aligned}$$

n n n n n n l ll
n n n n

5 The Bounded Case

n n N S,T,F,M_0,I n n N S,T,F,M_0
n l n n n n l s S
n l n l s^{cpl} S M_0 s M_0 s^{cpl} u s
u s > n s n N N n ll

Definition 5. N B,E,R, l
u B,E,R,l N s S e E

s l e u s b l^{-1} s^{cpl} b,e .
u N N

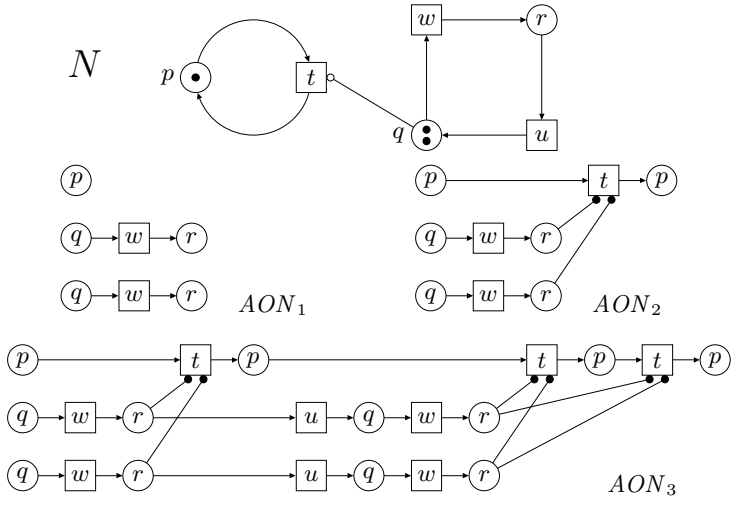
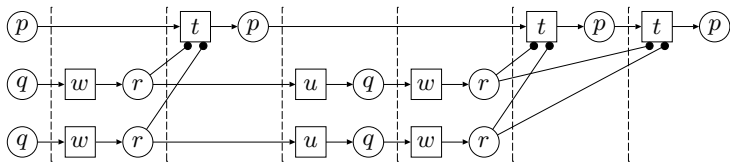


Fig. 5. i n N

l n n
l n n l n

Lemma 1. N N

n n n n τ n
n σ E_1 ... E_n τ l σ n
AON σ_{AON} AON n E E_1 ... E_n n n

$$\begin{array}{ccccccc}
& & l & n & N & n & \\
\tau & N & & & & n & n \quad B_0, \dots, B_n \\
n & & & l & n & & \\
\\
M_0 & l \ B_0 \ l \ E_1 \rangle_{N'} l \ B_1 \ \dots \ l \ E_n \rangle_{N'} l \ B_n \\
\\
& \tau & N & e & E_i & n & s \quad l \ e \\
n \ l \ B_{i-1} & s & l & l \ n & l \ B_{i-1} & s^{cpl} & u \ s \\
n & l & n & b & B & b, e & n \ l_b < i - \\
i & k_b & n & n & n & n & l \ l \ n \ l
\end{array}$$
$$\begin{array}{l}
\textbf{Definition 6.} \quad \tau \quad U_1 \dots U_n \quad u \quad N \\
\quad \quad \quad \tau \quad N_n \\
N_0, \dots, N_n \quad N_k \quad B_k, E_k, R_k, \quad k, l_k \quad k \quad n \quad u \quad u \\
- \quad N_0 \quad B_0, E_0, R_0, \quad 0, l_0 \quad 0 \\
- \quad m \quad k \quad N_k \quad B_k, E_k, R_k, \quad k, l_k \quad N_m \quad u \\
\quad B_m, E_m, R_m \quad l_m \quad m \\
\quad m \quad k \quad b, e \quad k \quad E_m \quad E_k \quad l_m \quad b^{cp l}, l \quad e \quad I \\
\quad u \quad \frac{ao}{\tau} \\
\quad \quad \quad \tau
\end{array}$$
[illegible]

6 Unboundedness

n n l n n n l n n n
 nn l n l n l n l n
 n n n n ll l
 ll l l l l n l n l n
 l ll n n k n u
 n ll n n N S, T, F, M_0, I n
 n

Definition 7.

$B, E, R, , l$ u B^n, E, R, l B^n E u z
 B^n B B^z B^z b B b, e R B E E B
 R R $(B^n$ $E)$ $(E$ $B^n)$ B^z E l
 u B E u u S_{aux} z E, aux, \sqsubset_{aux}
 E, R R E E R $, -1$ R E

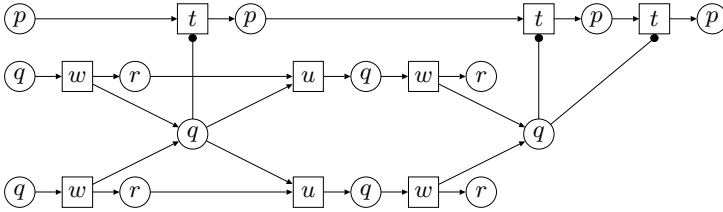


Fig. 7.

n l n n n n n n
 S z S_{aux} z n k n l n ll
 n l n n n n n l n l n
 n l n n l k n z Min z
 l l n n n ll n n l l
 n n n n n n n n n n
 n n n l

Definition 8.

z $B, E, R, , l$
 $- B$ B^n B
 B $b_{x,y}$ b B^z x b x b y b y b .
 $- R$ R $(B^n$ $E)$ $(E$ $B^n)$ $e, b_{e,y}$ e E $b_{x,e}, e$ e E
 $-$ $b_{x,y}, e$ b, e
 $- l$ B^n E l B^n E l $b_{x,y}$ l b b B
 ζ z $B, E, R, , l$ n n z

$$\begin{array}{ccccccc} n & l & n & n & z & n & n \\ & & n & n & l & n & n \\ n & n & n & n & l & ll & n \\ & z & \zeta & z & S & \zeta & z \\ n & nl & S & \sigma & \sigma & & N \\ n & n & n & n & n & & \end{array}$$

Definition 9. N z $B, E, R,$ l
 u

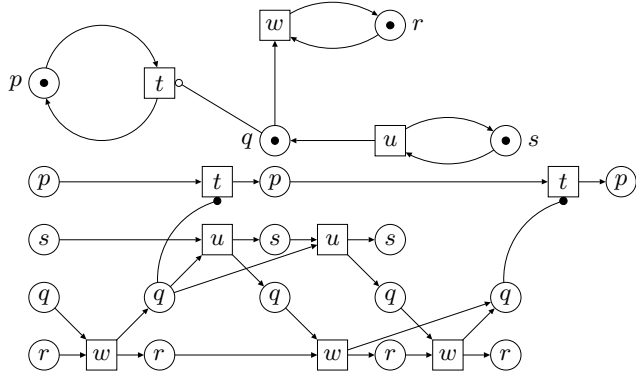


Fig. 8.

$$\begin{aligned}
 & - \begin{array}{ccccccc} N_0 & B_0, E_0, R_0, & 0, l_0 \\ E_0 & R_0 & 0 & B_0^z \\ B_0 & B_0^n & b_{s,i,0} & i & M_0 & s \\ l_0 & B_0 & S & u & l & b_{s,i,0} & s & b_{s,i,0} & B_0 \\ & 0 & B_0 \end{array} \\
 & - \begin{array}{ccccccc} m & k & N_k & B_k, E_k, R_k, & k, l_k & N_m & u \\ B_m^n & B_k^n & b_{s,t,i,m} & i & U_m & t & s & t \\ B_m^z & B_k^z & b_{s,m} & t & U_m & s & t & l_k & k & B_k^z \\ E_m & E_k & e_{t,i,m} & i & U_m & t & & & & \\ & & e_{t,i,m} & E_m & & & s & t \\ \widehat{b}_{s,t,i,m} & k & B_k^n & l^{-1} & s & & & & & \\ l_m & b_{s,t,i,m} & s & l_m & b_{s,m} & s & l_m & e_{t,i,m} & t & \\ & b_{s,t,i,m} & B_m^n & B_k^n & b_{s,m} & B_m^z & B_k^z & e_{t,i,m} & E_m & E_k \\ l_m & x & l_k & x & x & B_k & E_k & & & \\ R_m & R_k & \left(\begin{array}{cccccc} \widehat{b}_{s,t,i,m}, e_{t,i,m} & e_{t,i,m} & E_m & s & t \\ e_{t,i,m}, b_{s,t,i,m} & e_{t,i,m} & E_m & s & t \end{array} \right) & R_m & R_m \end{array} \\
 & R_m \left\{ \begin{array}{cccc|cccc} e, b_{s,m} & E_k & B_m^z & B_k^z & s, l_k & e & F & b & B_k^z \\ & & & & l_k & b & s & e, b & R_k \end{array} \right\} \\
 & R_m \left\{ \begin{array}{cccc|cccc} b_{s,i}, e & B_m^z & E_m & E_k & l_m & e, s & F & i \\ & & & & b_{s,j} & B_m^z & j & \end{array} \right\}. \\
 & \begin{array}{ccccccccccc} m & k & b, e & Max_m & B_m^z & E_m & E_k & l_m & b, l_m & e & I \\ & m & b & B_m & e & E_m & b, e & R_m & & & \end{array} \\
 & u \quad \begin{array}{c} zao \\ \tau \end{array} \quad \tau \\
 & \begin{array}{ccccccc} n & n & ll & n & n \\ \tau & w, w & t & u, u & w, w & t & t & n \\ n & n & nn & n & n & n & n \end{array}
 \end{aligned}$$

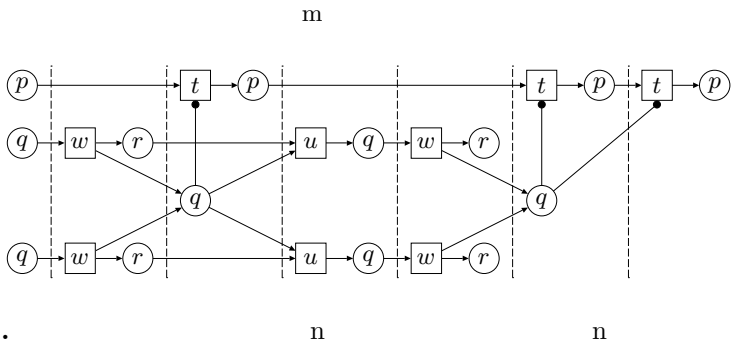
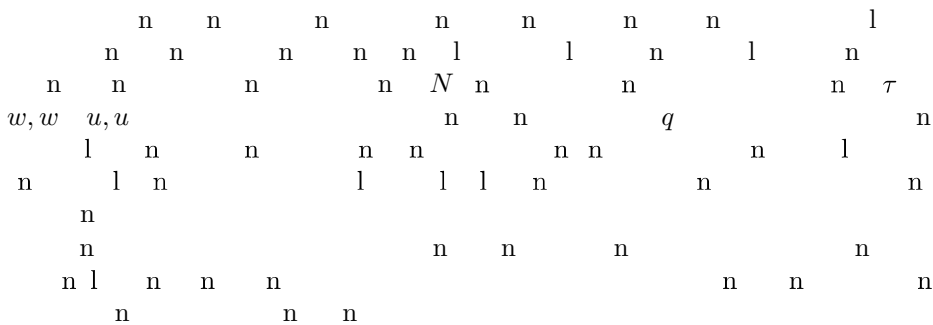


Fig. 9.



Theorem 4.

$$\begin{aligned}
 & \overset{z}{N} \quad \bigcup_{\tau \text{ steps}(N)} \overset{zao}{\tau} \\
 & \overset{N}{\tau} \quad \bigcup_{AON^z} \overset{aon^z(N)}{N} \overset{lsteps}{\tau} \quad \overset{z}{S} \quad \overset{z}{\tau}
 \end{aligned}$$

Fig. 10.

7 Concluding Remarks

The main result of this paper is the construction of a new model of computation, the *distributed Petri net*, which is a natural extension of the classical Petri net model. This model is able to capture the essential features of distributed systems, such as the local nature of the computation and the asynchronous communication between the processes. The distributed Petri net model is a powerful tool for the analysis and synthesis of distributed systems. It allows us to study the properties of distributed systems in a systematic and rigorous way. The distributed Petri net model is a natural extension of the classical Petri net model, and it is a powerful tool for the analysis and synthesis of distributed systems.

Acknowledgment

The author would like to thank the anonymous referees for their valuable comments and suggestions.

References

- [1] A. A. Chaitin, *Theoretical Computer Science*, vol. 1, pp. 1-10, 1975.
- [2] M. P. Heule, *Nonsequential Processes. A Petri Net View*, pp. 1-10, 1985.
- [3] M. P. Heule, *Fundamenta Informaticae*, vol. 1, pp. 1-10, 1985.
- [4] M. P. Heule, *Theory of Relations*, vol. 1, pp. 1-10, 1985.
- [5] M. P. Heule, *LICS'87*, pp. 1-10, 1987.
- [6] M. P. Heule, *Information and Control*, vol. 1, pp. 1-10, 1987.
- [7] M. P. Heule, *Theoretical Computer Science*, vol. 1, pp. 1-10, 1987.
- [8] M. P. Heule, *Information and Computation*, vol. 1, pp. 1-10, 1987.
- [9] M. P. Heule, *Acta Informatica*, vol. 1, pp. 1-10, 1987.
- [10] M. P. Heule, *Fundamenta Informaticae*, vol. 1, pp. 1-10, 1987.
- [11] M. P. Heule, *Distributed Computing*, vol. 1, pp. 1-10, 1987.

1 *Petri Net Theory and the Modeling of Systems* ll)
1 1 1 1 *Fundamenta Mathematicae*
)
1 1 1 m m *Advances in Petri*
Nets. Lectures on Petri Nets I: Basic Models)
1 1 m m)
1 1 m *MFCS'97*
) 1 m

Improved Question-Guided Stubborn Set Methods for State Properties

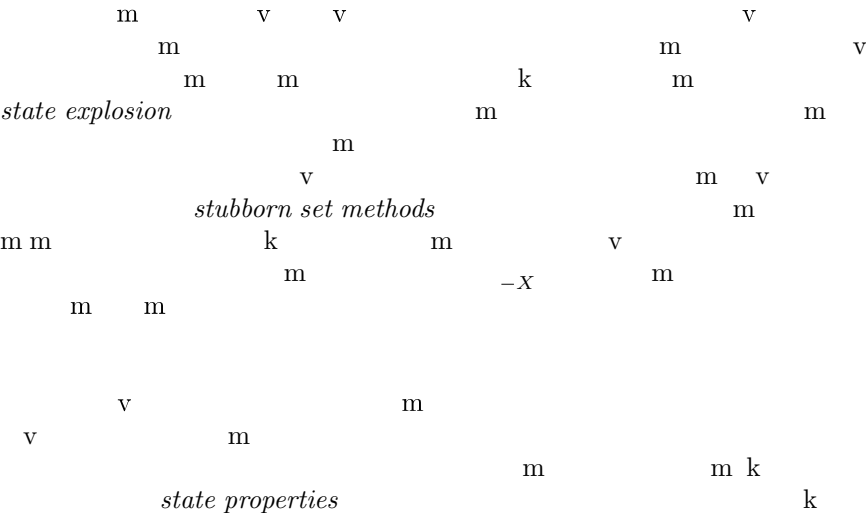
1 m 2

¹ University of Aarhus, Department of Computer Science
Aabogade 34, DK-8200 Aarhus N., Denmark
kris@daimi.au.dk

² Tampere University of Technology, Software Systems Laboratory
PO Box 553, FIN-33101 Tampere, Finland
ava@cs.tut.fi

Abstract. We present two new question-guided stubborn set methods for state properties. The first method makes it possible to determine whether a marking is reachable in which a given state property holds. It generalises the results on stubborn sets for state properties recently suggested by Schmidt in the sense that his method can be seen as an implementation of our more general method. We propose also alternative, more powerful implementations that have the potential of leading to better reduction results. This potential is demonstrated on some practical case studies. As an extension of the first method, we present a second method which makes it possible to determine if from all reachable markings it is possible to reach a marking where a given state property holds. The novelty of this method is that it does not rely on ensuring that no transition is ignored in the reduced state space. Again, the benefit is in the potential for better reduction results.

1 Introduction



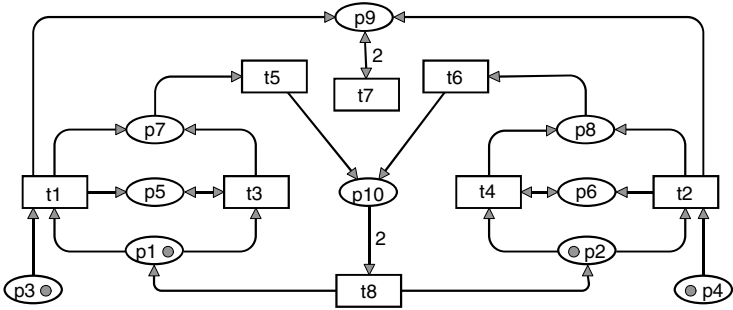


Fig. 1. m

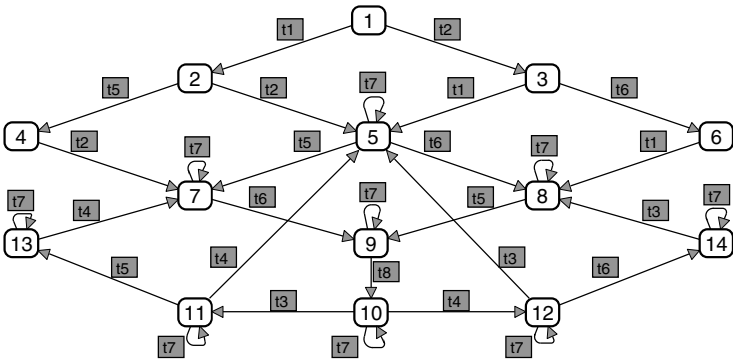
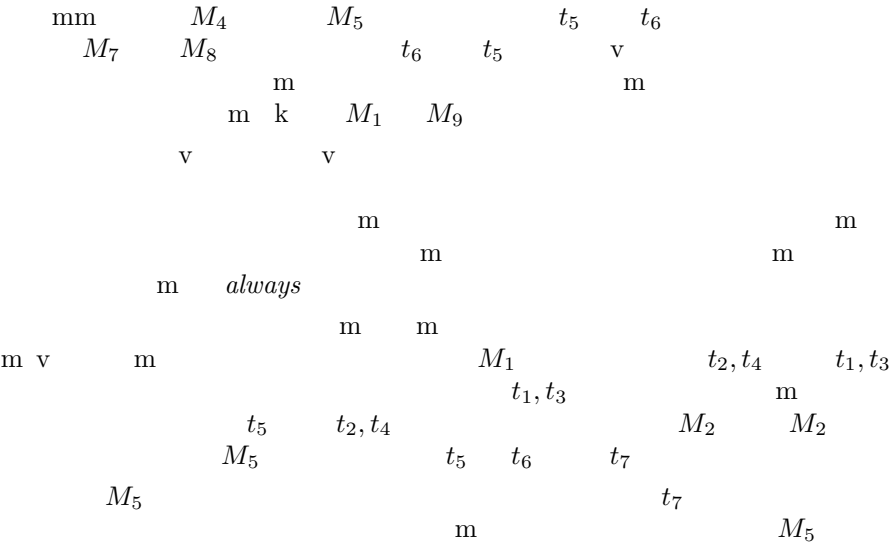
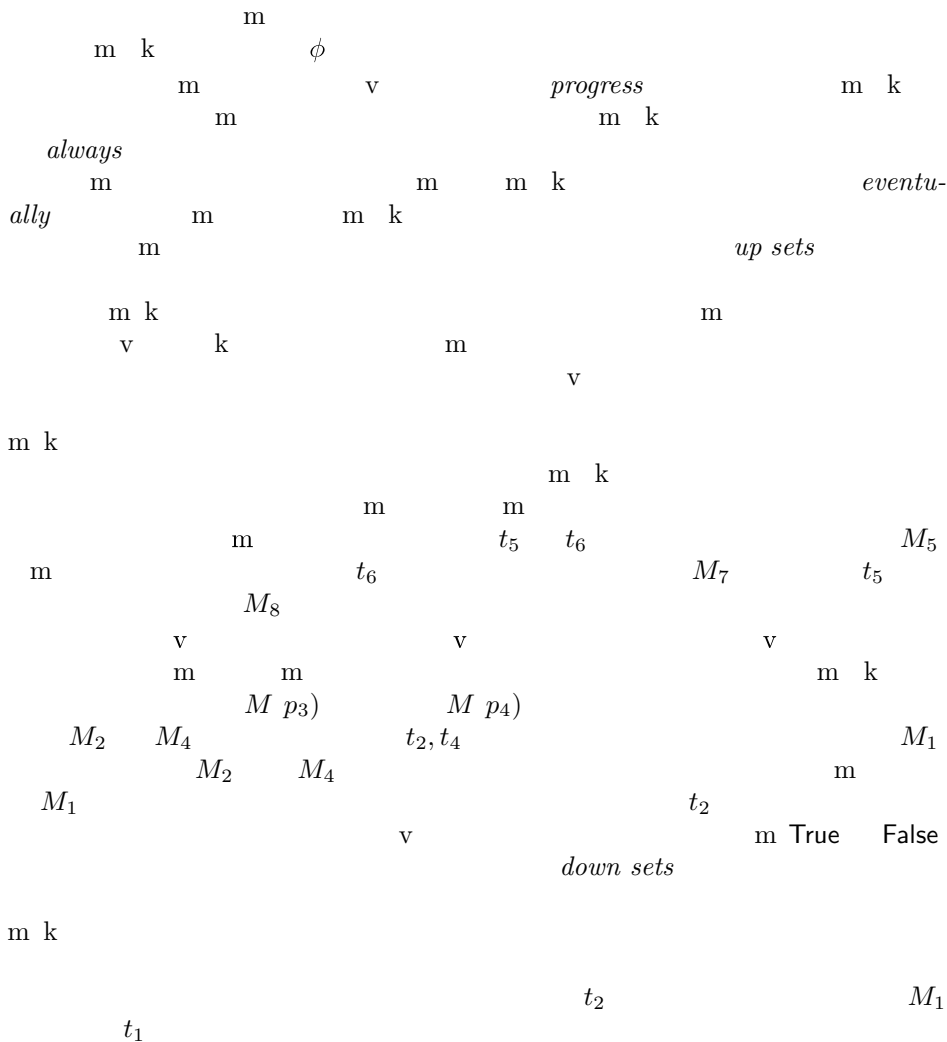
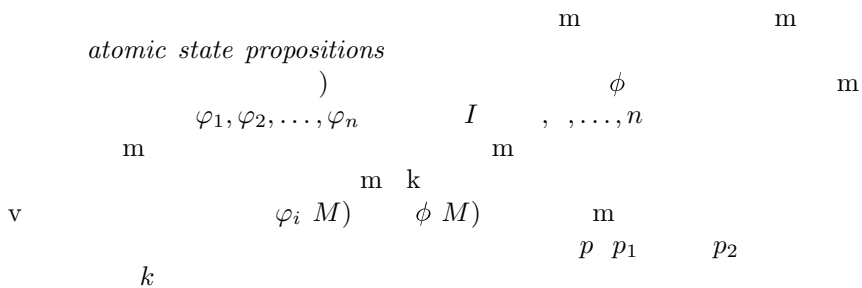


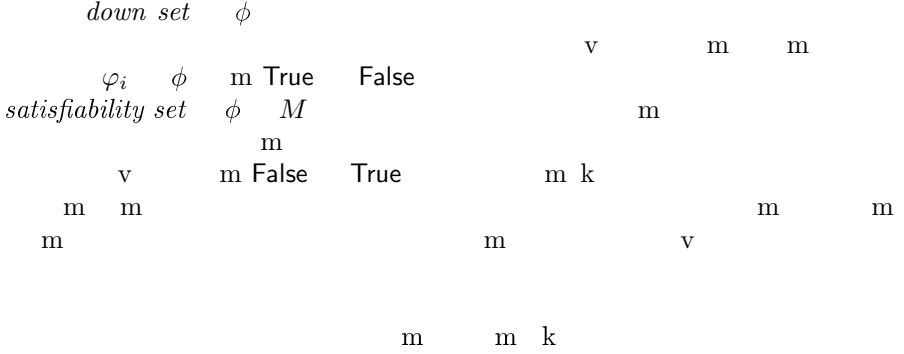
Fig. 2.





4 State Properties





Definition 4. Let ϕ be a state property constructed from the atomic state propositions φ_i $i \in I$ and let $M_0 \in M_I$. A set of transitions $T \subseteq T$ has the **up set property** in M_0 with respect to ϕ iff the following holds for all occurrence sequences $M_0 \xrightarrow{t_1 t_2 \dots t_n} M_n$ starting in M_0 :

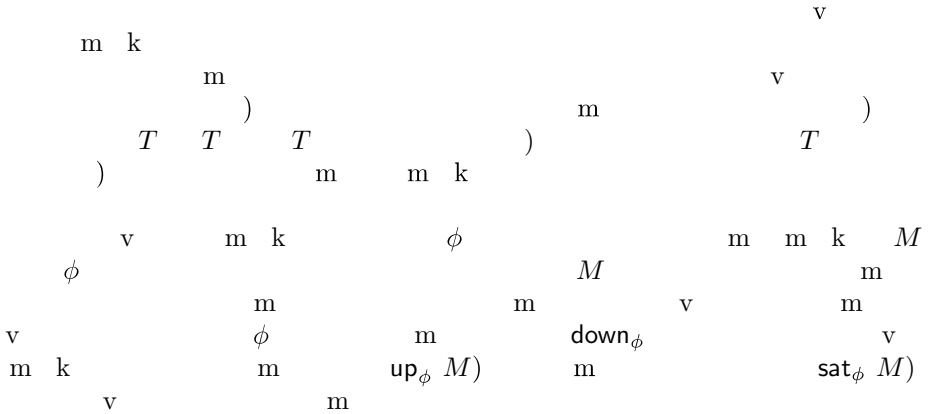
$$\phi(M_0) \wedge \phi(M_n) \implies \exists j \in J \quad \exists t_j \in T$$

A set of transitions $T \subseteq T$ has the **down set property** with respect to ϕ iff the following holds for all markings $M, M' \in M_I$, all $t \in T$, and all $i \in I$:

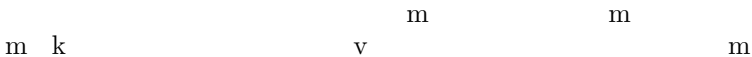
$$M \xrightarrow{t} M' \wedge \varphi_i(M) \wedge \varphi_i(M') \implies t \in T$$

A set of indices $J \subseteq I$ has the **satisfiability set property** in M_0 with respect to ϕ iff the following holds for all occurrence sequences $M_0 \xrightarrow{t_1 t_2 \dots t_n} M_n$:

$$\phi(M_0) \wedge \phi(M_n) \implies \exists i \in J \quad \varphi_i(M_0) \wedge \varphi_i(M_n) \quad \square$$



6 Preserving Reachability of State Properties



$$\frac{m}{v} \quad m$$

Definition 5. Let M be a marking and ϕ a state property constructed from the atomic state propositions φ_i $i \in I$. A set $T_\phi(M) \subseteq T$ is **Reachability of a State Property Preserving (RSPP) stubborn** in M , iff the following hold:

D1 If $t_1, \dots, t_n \in T_s(M)$, $t \in T_s(M)$, $M \models t_1 t_2 \dots t_n \rangle M_n$, and $M_n \models t \rangle M_n$, then there is M' such that $M' \models t \rangle M'$ and $M' \models t_1 t_2 \dots t_n \rangle M_n$.

SPP1 If $\phi \models M$ and $t \models M \mid t \rangle$ $t \models \text{down}_\phi \mid t \rangle$ $T_s(M)$ then $\text{up}_\phi \mid M \rangle$ $T_s(M)$.

SPP2 For every $i \in \text{sat}_\phi(M)$ there is an occurrence sequence $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \dots \xrightarrow{t_n} M_n$ such that $M = M_0$, t_j is a key transition of $T_s(M_{j-1})$ for $j = 1, \dots, n$, and $\phi(M_n) \uparrow_{\varphi_i} M_n$. \square

[illegible]

Lemma 1. *Let ϕ be a state property, $SG(V, E)$ the full state space, and $SSG(V_{SSG}, E_{SSG})$ an SS state space constructed using RSPP-stubborn sets. Let $M_0 \in V_{SSG}$ be a marking such that $\phi(M_0)$ and for which there exists an occurrence sequence $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \dots \xrightarrow{t_n} M_n$ such that $\phi(M_n)$ holds. Then there is a marking $M_0' \in M_0 \rangle_{SSG}$ such that the following holds.*

1. There are transitions t_1, t_2, \dots, t_m and markings M_1, M_2, \dots, M_m such that $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \dots \xrightarrow{t_m} M_m$ and $\phi(M_m)$ holds.
2. The occurrence sequence in item 1 leading to a marking where ϕ holds is no longer than the original occurrence sequence, i.e., $m \leq n$.

3. The length of the occurrence sequence in item 1 has decreased, i.e., $m < n$, or the set of the atomic state propositions of ϕ which are satisfied has grown, i.e., $\text{on}_\phi M_0) \quad \text{on}_\phi M_0)$. \square

Proof. $\phi M_0) \quad \phi M_n) \quad \text{sat}_\phi M_0) \quad i \quad \varphi_i M_0)$
 $\varphi_i M_n) \quad \phi M_0) \quad T_s M_0) \quad k \quad t_1, \dots, t_k \quad m \quad k \quad M_0)$
 $M_0, \dots, M_k \quad V_{SSG} \quad M_0 \quad M_0 t_1 \rangle_{SSG} M_1 t_2 \rangle_{SSG} \quad t_k \rangle_{SSG} M_k$
 $\phi M_k) \quad \text{up}_{\varphi_i} M_k) \quad \begin{matrix} k \\ j=0 \end{matrix} T_s M_j) \quad \begin{matrix} m \\ j \end{matrix} \quad \begin{matrix} k \\ j \end{matrix} \quad \begin{matrix} m \\ j \end{matrix} \quad M_0 \quad M_j$
 $\phi M_j) \quad \begin{matrix} m \\ j \end{matrix} \quad \begin{matrix} m < n \\ j, \end{matrix} \quad \begin{matrix} k \\ j \end{matrix} \quad \phi M_0) \quad \phi M_n) \quad m \quad m$
 $m \quad m \quad j, \quad j \quad k \quad \phi M_j)$

(1) $t_1, \dots, t_n \quad T_s M_h) \quad v \quad h < l \quad k$
 $k \quad t_1, \dots, t_k \quad m \quad k \quad M_0, \dots, M_l$
 $M_n \quad M_0 t_1 \quad t_l \rangle M_l \quad M_l t_1 \quad t_n \rangle M_l \quad m$
 $m \quad m \quad h \quad t_{h+1} \quad \text{down}_\phi$
 $\phi M_0), \dots, \phi M_h) \quad \phi M_h)$
 $\phi M_h) \quad m \quad t_1, \dots, t_n \quad \text{up}_\phi M_h) \quad T_s M_h)$
 $m \quad t_1, \dots, t_l \quad \text{down}_\phi$
 $v \quad \text{on}_\phi M_0) \quad \text{on}_\phi M_1) \quad \text{on}_\phi M_l) \quad \text{on}_\phi M_0) \quad \text{on}_\phi M_1)$
 $\text{on}_\phi M_l) \quad \phi M_0), \dots, \phi M_l)$

Case A: $t_1, \dots, t_n \quad T_s M_j) \quad m \quad j \quad k$
 $m \quad j \quad) \quad l \quad j \quad T_s M_j) \quad k$
 $t_1, t_2, \dots, t_n \quad k \quad t_h$
 $M_j t_1 \quad t_{h-1} t_h t_{h+1} \quad t_n \rangle M_j \quad m \quad k \quad M$
 $M_j t_h \rangle M \quad t_1 \quad t_{h-1} t_{h+1} \quad t_n \rangle M_j \quad m \quad M_0$
 $M \quad m \quad n -$

Case B: $t_1, \dots, t_n \quad T_s M_j) \quad v \quad j \quad k \quad) \quad v$
 $\text{on}_\phi M_0) \quad \text{on}_\phi M_k) \quad \text{on}_\phi M_0) \quad \text{on}_\phi M_k) \quad \text{on}_\phi M_0) \quad \text{on}_\phi M_k)$
 $m \quad v \quad t_1, t_2, \dots, t_n \quad \text{up}_{\varphi_i} M_k) \quad \begin{matrix} k \\ j=0 \end{matrix} T_s M_j) \quad m \quad t_1, \dots, t_n \quad T_s M_j)$
 $v \quad j \quad k \quad \text{on}_\phi M_0) \quad \text{on}_\phi M_k) \quad m$
 $M_0 \quad M_k \quad m \quad n$

$m \quad m \quad m \quad k \quad m \quad k \quad m \quad k \quad m$
 $mm \quad m \quad m \quad M_0 \quad M_l$

Theorem 1. Let ϕ be a state property, $SG(V, E)$ be the full state space, $SSG(V_{SSG}, E_{SSG})$ an SS state space constructed using RSPP-stubborn sets, and let $M_0 \in V_{SSG}$. Then:

$$M \models M_0 \rangle \phi \Leftrightarrow M \models M_0 \rangle_{SSG} \phi \quad \square$$

Proof. $m \in V_{SSG} \cap V \cap E_{SSG} \cap E$
 $M_0 \rangle_{SSG}, M_2 \models M_0 \rangle_{SSG}, \dots$ $M_n \models M_0 \rangle_{SSG}$ $M_1 \models \phi(M_n)$
 $M_\phi^0 \models M$ σ_i $M_i \models \sigma_i \rangle M_\phi^i$ $\phi(M_\phi^i)$ M_ϕ^i
 $M_\phi \models V$ σ $\text{distance } \Delta(M, \sigma, \phi)$ $m \models k$ $M \models V_{SSG}$ $m \models k$
 $\Delta(M, \sigma, \phi) = I \Rightarrow \sigma \text{ off}_\phi(M)$
 $m \models k$
 $m \models k$ ϕ $m \models m$ m
 $m \models \phi$ M_{i-1} $m \models k$ M_ϕ^i $M_i \models \sigma_i \rangle M_\phi^i$ $M_{i-1} \rangle_{SSG}$
 $M_\phi^i \models m$ $\Delta(M_i, \sigma_i, \phi) < \Delta(M_{i-1}, \sigma_{i-1}, \phi)$
 $M_n \models M_0 \rangle_{SSG}$ ϕ

7 Preserving Home State Properties

$m \models k$ $m \models m$ $m \models v$ M
 $M_I \rangle M$ $M \rangle \phi(M)$ $m \models m$ M
 $m \models \text{not}$ $m \models k$ v $\phi(M)$
 $\phi(M) \rangle \phi(M)_{SSG}$ v $M \rangle_{SSG} \phi(M)$
 $m \models m$ m

Definition 6. Let M be a marking and ϕ a state property. A set $T_s(M)$ is **Home State Property Preserving (HSPP) Stubborn** in M , iff $T_s(M)$ is RSPP stubborn in M and the following hold:

D2 If $t_1, \dots, t_n \in T_s(M)$, $t \in T_s(M)$, $M \models t_1 t_2 \dots t_n \rangle M_n$, and $M \models t$, then $M_n \models t$.

SPP3 For every $t \in \text{down}_\phi$ there is an occurrence sequence $M_0 \models t_1 \rangle \dots \rangle t_n \rangle M_n$ such that $M \models M_0, t_j \in T_s(M_{j-1})$ for $j = 1, \dots, n$, and $t \in T_s(M_n)$. \square

$v \in T_s(M)$

$m \in m \in m \in k \in \phi$

$k \in m \in k \in m \in \phi \in m \in)$

$k \in m \in k \in T_s(M)$

$m \in m \in$

strong stubborn sets *ensure*

$v \in m \in m \in k \in m \in \phi \in m$

$m \in m \in k \in \phi$

$m \in m \in mm \in m$

$m \in M_0 \in M_I$

Theorem 2. Let ϕ be a state property, $SG = (V, E)$ be the full state space, $SSG = (V_{SSG}, E_{SSG})$ an SS state space constructed using HSPP stubborn sets, and let $M_0 \in V_{SSG}$. Then:

$$M \models M_0 \rangle \phi / M \rangle \Leftrightarrow M_{SSG} \models M_0 \rangle_{SSG} \phi / M_{SSG} \rangle_{SSG} \quad \square$$

Proof. $mm \in m \in m \in v \in v$

n

$$(1) \quad M^n \models M_0^n \models t_1^n, \dots, t_n^n \rangle M^n \quad M_0^n \models M_0 \rangle_{SSG} M_0^n \models t_1^n t_2^n \dots t_n^n \rangle$$

$$\begin{aligned} & \quad) \quad m \quad v \quad v \quad n \quad m \\ & \quad m \quad M_0^n \quad M_0 \quad \phi / M_0^n \rangle \quad \phi / M_0^n \rangle_{SSG} \\ & \quad M_0^n \quad M_{SSG} \quad m \\ & \quad) \quad m \quad n \quad \phi \quad M_0^n \rangle \quad) \quad m \quad m \\ & \quad m < n \\ & \quad m \quad \phi \quad M_0^n \rangle \quad m \quad t_1, \dots, t_h \\ & \quad M_0, \dots, M_h \quad M_0^n \quad M_0 \quad M_0 \models t_1 \rangle_{SSG} M_1 \models t_2 \rangle_{SSG} \dots t_h \rangle_{SSG} M_h \\ & \quad \phi \models M_h \rangle \quad M_0 \models t_1 \rangle_{SSG} M_1 \\ & \quad t_2 \rangle_{SSG} \dots t_k \rangle_{SSG} M_k \quad M_0^n \quad M_0 \quad t_1^n, \dots, t_n^n \quad T_s(M_j) \\ & \quad m \quad j \quad k \end{aligned}$$

Case A: $t_1^n, \dots, t_n^n \in T_s(M_0) \quad T_s(M_h)$

$j \quad k \quad h \quad t_i \quad t_i \quad M_i \quad M_i \quad i \quad h$

Case B: t_1^n, \dots, t_n^n $T_s M_0)$ $T_s M_h)$
 M_h $M_h t_1^n t_n^n M_h$ $M^n t_1 t_h M_h$
 $\phi / M^n \rangle$ k $\phi M_h)$ $\phi M_h)$
 t_i^n t_1^n, \dots, t_n^n t_i^n down_ϕ v

m j t_1^n, \dots, t_n^n $T_s M_j)$ i
 m m $t_i^n T_s M_j)$ M^{n-1}
 $M_j t_1^n t_n^n M^{n-1}$ $M^n t_1 t_j M^{n-1}$ $v \phi / M^{n-1} \rangle$
 $\phi / M^n \rangle$ M_0^{n-1}
 $M_j t_i^n \rangle_{SSG} M_0^{n-1} t_1^n t_{i-1}^n t_{i+1}^n t_n^n M^{n-1}$ v $)$ $n -$

8 Implementation

m m m
 v
 m

8.1 Implementation of Up/Down and Satisfiability Sets

v

v

Definition 7. Let M be a marking and ϕ a state property. The up set $\text{up}_\phi M)$ in M is defined as follows. If ϕ holds in M we define $\text{up}_\phi M)$. If ϕ does not hold in M we define $\text{up}_\phi M)$ according to the following cases:

Case $\phi M p) = k$: $\text{up}_\phi M)$ consists of the transitions which can remove tokens from p and which add at most k tokens to p :

$$\text{up}_\phi M) = \{ t \mid T(W(p, t) > W(t, p) - W(t, p)) = k \}$$

Case $\phi M p) = k$: If p contains too few tokens then $\text{up}_\phi M)$ consists of the transitions which can add tokens and do not require additional tokens to be present on p , and if p contains too many tokens then $\text{up}_\phi M)$ consists of the transitions which can remove tokens from p and which add at most k tokens:

$$\text{up}_\phi M) = \begin{cases} \{ t \mid T(W(p, t) < W(t, p) - W(p, t) - M(p)) = k & \text{if } M(p) < k \\ \{ t \mid T(W(p, t) > W(t, p) - W(t, p)) = k & \text{if } M(p) > k \end{cases}$$

Case $\phi M p) = k$: $\text{up}_\phi M)$ consists of the transitions which can add tokens and which do not require additional tokens to be present on p :

$$\text{up}_\phi M) = \{ t \mid T(W(p, t) < W(t, p) - W(p, t) - M(p)) = k \}$$

Case $\phi \ M \ p) \ k : \text{up}_\phi M)$ consists of the transitions which can change the marking of p and which do not require additional tokens to be present on p :

$$\text{up}_\phi M) \quad t \quad T \ W \ p, t) \quad W \ t, p) \quad W \ p, t) \quad k$$

Case $\phi \ M \ p_1) > M \ p_2) \text{ or } \phi \ M \ p_1) \ M \ p_2) :$

$$\text{up}_\phi M) \quad t \quad T \ W \ t, p_1) - W \ p_1, t) > W \ t, p_2) - W \ p_2, t)$$

Case $\phi \ M \ p_1) \ M \ p_2) : \text{up}_\phi M)$

$$\left\{ \begin{array}{ll} t \quad T \ W \ t, p_1) - W \ p_1, t) > W \ t, p_2) - W \ p_2, t) & \text{if } M \ p_1) < M \ p_2) \\ t \quad T \ W \ t, p_1) - W \ p_1, t) < W \ t, p_2) - W \ p_2, t) & \text{if } M \ p_1) > M \ p_2) \end{array} \right.$$

Case $\phi \ M \ p_1) \ M \ p_2) :$

$$\text{up}_\phi M) \quad t \quad T \ W \ t, p_1) - W \ p_1, t) \quad W \ t, p_2) - W \ p_2, t)$$

Case $\phi \ \phi_1 \ \phi_2 : \text{up}_\phi M)$ is the up set of one of the ϕ_i s which does not hold in M :

$$(\phi_1 M) \text{ up}_\phi M) \text{ up}_{\phi_1} M) \quad (\phi_2 M) \text{ up}_\phi M) \text{ up}_{\phi_2} M)$$

Case $\phi \ \phi_1 \ \phi_2 : \text{up}_\phi M) \text{ up}_{\phi_1} M) \text{ up}_{\phi_2} M)$ □

Definition 8. Let ϕ be a state property. The down set down_ϕ is defined as follows:

Case $\phi \ M \ p) \ k : \text{down}_\phi \quad t \quad T \ W \ p, t) < W \ t, p) \quad W \ p, t) \quad k$

Case $\phi \ M \ p) \ k : \text{down}_\phi \quad t \quad T \ W \ p, t) \quad W \ t, p) \quad W \ p, t) \quad k$

Case $\phi \ M \ p) \ k : \text{down}_\phi \quad t \quad T \ W \ p, t) \quad W \ t, p) \quad W \ t, p) \quad k$

Case $\phi \ M \ p) \ k : \text{down}_\phi \quad t \quad T \ W \ p, t) > W \ t, p) \quad W \ t, p) < k$

Case $\phi \ M \ p_1) > M \ p_2) \text{ or } \phi \ M \ p_1) \ M \ p_2) :$

$$\text{down}_\phi \quad t \quad T \ W \ p_1, t) - W \ t, p_1) > W \ p_2, t) - W \ t, p_2)$$

Case $\phi \ M \ p_1) \ M \ p_2) \text{ or } \phi \ M \ p_1) \ M \ p_2) :$

$$\text{down}_\phi \quad t \quad T \ W \ t, p_1) - W \ p_1, t) \quad W \ t, p_2) - W \ p_2, t)$$

Case $\phi \ \phi_1 \ \phi_2 \text{ or } \phi \ \phi_1 \ \phi_2 : \text{down}_\phi \quad \text{down}_{\phi_1} \quad \text{down}_{\phi_2}$ □

Definition 9. Let M be a marking and ϕ a state property constructed from the atomic state propositions $\varphi_i \ i \ I$. The satisfiability set $\text{sat}_\phi M)$ in M is defined as follows. If $\phi M)$ holds then $\text{sat}_\phi M) = I$. Otherwise it is defined as follows.

Case $\phi \ \varphi_i : \text{sat}_\phi M) = i$

Case $\phi \ \phi_1 \ \phi_2 : \text{sat}_\phi M) = \text{sat}_{\phi_1} M) \cup \text{sat}_{\phi_2} M)$

Case $\phi \ \phi_1 \ \phi_2 : \text{sat}_\phi M)$ is the satisfiability set of one of the ϕ_i s which does not hold in M :

$$\phi_1 M) \cup \text{sat}_\phi M) \cup \text{sat}_{\phi_1} M)) \cup \phi_2 M) \cup \text{sat}_\phi M) \cup \text{sat}_{\phi_2} M))$$

□

V V

Proposition 3. *Let M be a marking and assume that $\text{up}_\phi(M) \rightarrow T$, $\text{down}_\phi(T) \rightarrow T$, and $\text{sat}_\phi(M) \rightarrow I$ are constructed according to Def. 7, Def. 8, and Def. 9, respectively. Then the following hold:*

1. $\text{up}_\phi M$) has the up set property in M with respect to ϕ .
2. down_ϕ has the down set property with respect to ϕ .
3. $\text{sat}_\phi M$) has the satisfiability set property in M with respect to ϕ . \square

8.2 Implementation of RSPP and HSPP Stubborn

Attractor Set.

M

$v \quad i \quad \text{sat}_\phi(M)$

n

m

M

$i \quad \text{sat}_\phi(M) \quad \text{up}_{\varphi_i(M)} \quad T_s(M)$

$\text{down}_\phi \quad T_s(M)$

$\text{strong stubborn sets}$

TARJAN

C

M_0

$\phi(M)$ m M C m φ_i v
 $\text{up}_{\varphi_i}(M_0)$ M C $T_s(M)$ m
 $) \text{up}_{\varphi_i}(M_0)$ M C $T_s(M)$ $)$ M_0 $T_s^{\text{up}_i}(M_0)$
 $\text{up}_{\varphi_i}(M_0) -$ M C $T_s(M)$
 $T_s(M_0)$ M_0 $T_s^{\text{up}_i}(M_0)$
 m m m
 t down_{ϕ} M C t $T_s(M)$ k v
 m m m v m

m m

Cycle Detection. m
 m m
 m m TARJAN m
 m
 v m k M_1
 k v k
 v k m k M_n, M_{n-1}, \dots, M_1 k i
 $\text{up}_{\varphi_i}(M_1)$ $\text{up}_{\varphi_i}(M_1)$ $\text{up}_{\varphi_i}(M_1) -$ $\text{up}_{\varphi_i}(M_1)$ M_1 $T_s^{\text{up}_i}(M_1)$
 $T_s^{\text{up}_i}(M)$ k down_{ϕ} k

9 Applications

v m *boundedness properties*
 m m
 v v v
 v m
 v m m
 m
Best Upper Bounds. k *upper bound* P P
 M M_I p P' M p k m m
 k *best upper bound* P k
 $\phi^k(M)$ p P' M p k m k
 k_0 m k p P' M p k_0 v
 $k_0 -$ m

$$\begin{array}{ccccccc} & k & & v & & p & P' & M & p) & k & & m \\ & & v & & & m & k & & & & & m \\ k & & & & & & & & & & & m \\ & & & & \phi^k & & & & & & & \\ & k & & P & & & & & k & & & \\ P & & & & & & & & & m & v & k \\ m & P & & & & k & k & & P & & m & \\ \text{up}_{\phi^k} & M) & t & T & & W & p, t) < & W & t, p) & & W & p, t) & M & p) \\ & & & & p & P' & & p & P' & & p & P' & & p & P' \\ \text{down}_{\phi^k} & & t & T & & W & p, t) > & W & t, p) & & W & t, p) < & k \\ & & & & p & P' & & p & P' & & p & P' & & \\ & & v & & v & & \text{up}_{\phi^k} & & k & & \text{down}_{\phi^k} \\ m & & m & & v & & m & v & p & P' & W & t, p) < & k & m \\ m & & & & & & & & & & & k & & \\ & & j & & & & & & & & & & & \end{array}$$

$$\begin{array}{ccccccc} \textit{Best Lower Bounds.} & & k & & \textit{lower bound} & & P & P \\ M & M_I) & p & P' & M & p) & k & m & m \\ k & & & & \textit{best lower bound} & & P & m \\ & & & & & & m & \phi^k & M) & p & P' & M & p) & k \\ & k & & & k_0 & & k_0 & & & m & k \\ p & P' & M & p) & & & & & & p & P' & M & p) & k \\ & & & m & & & & & & & & & & \\ & & & & & & & & & p & P' & M & p) & k \\ & & & & & & & & & & & & & k \\ m & & k & & & & & & m & & & & & \\ & & m & & & & & & k & & & & & \end{array}$$

$$\begin{array}{ccccccc} \text{up}_{\phi^k} & M) & t & & W & p, t) > & W & t, p) & & W & t, p) & k \\ & & & & p & P' & & p & P' & & p & P' \\ \text{down}_{\phi^k} & & t & & W & p, t) < & W & t, p) & & W & p, t) & k \\ & & & & p & P' & & p & P' & & p & P' \end{array}$$

10 Experimental Results

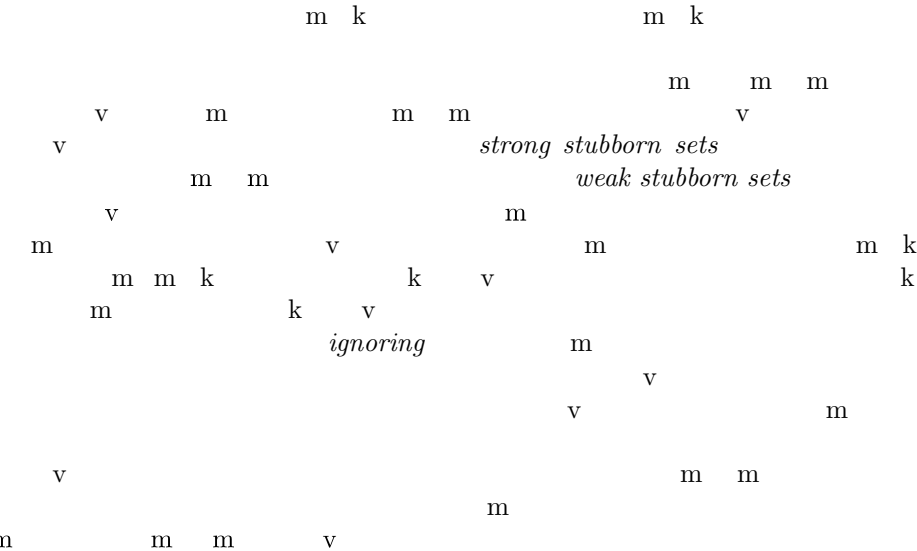
$$\begin{array}{ccccccc} v & m & m & & & m & \\ & & & & m & m & \textit{Attractor Set} & \textit{Cycle} \\ \textit{Detection} & & m & v & & & & \\ & & \textit{strong component algorithm} & & & & & k \end{array}$$

$$\begin{array}{ccccccc} & & v & m & & & \\ m & m & m & m & m & \text{PETERSON} & \text{HYMAN} & m \\ & & m & & m & & \textit{Mutual Excl.}) \end{array}$$



Table 2. m m

Model/ Property	Full State Space		Attractor Set Method					
			DFG		BFG		CG	
	Nodes	Arcs	Nodes	Arcs	Nodes	Arcs	Nodes	Arcs
Peterson	58	116						
Reach. of CS			9	8	10	11	39	67
Mutual Excl.			-	-	-	-	50	90
Hyman	80	160						
Reach. of CS			7	6	14	17	60	95
Mutual Excl.			36	42	49	76	64	106
Reader/Writer	136	532						
Reach. of Write			3	2	3	2	77	221
Reach. of Read			3	2	3	2	85	197
Excl. Write			-	-	-	-	118	419
Master/Slave								
DoneIdle-3	232	588	61	79	212	520	229	548
DoneIdle-5	7,840	32,656	1,623	4,144	7,700	31,966	7,837	32,412
DoneIdle-6	46,784	233,856	9,819	37,155	33,092	156,317	46,701	233,276
DoneWIdle-3	232	588	-	-	-	-	231	562
DoneWIdle-5	7,840	32,656	-	-	-	-	7,839	32,494
DoneWIdle-6	46,784	233,856	-	-	-	-	46,783	233,470



References

[1] S. Christensen, J. B. Jørgensen, and L. M. Kristensen. Design/CPN - A Computer Tool for Coloured Petri Nets. In *Proceedings of TACAS'97*, volume 1217 of *Lecture Notes in Computer Science*, pages 209–223. Springer-Verlag, 1997.

[2] A. Gibbons. *Algorithmic Graph Theory*. Cambridge University Press, 1985.

[3] P. Godefroid. Using Partial Orders to Improve Automatic Verification Methods. In *Proceedings of Computer-Aided Verification '90*, volume 531 of *Lecture Notes in Computer Science*, pages 175–186. Springer-Verlag, 1990.

- [4] P. Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems, An Approach to the State-Explosion Problem*, volume 1032 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.
- [5] D. Peled. All from One, One for All: On Model Checking Using Representatives. In *Proceedings of Computer-Aided Verification '93*, volume 697 of *Lecture Notes in Computer Science*, pages 409–423. Springer-Verlag, 1993.
- [6] K. Schmidt. Stubborn Sets for Standard Properties. In *Proceedings of ICATPN'99*, volume 1639 of *Lecture Notes in Computer Science*, pages 46–65. Springer-Verlag, 1999.
- [7] A. Valmari. Error Detection by Reduced Reachability Graph Generation. In *Proceedings of the 9th European Workshop on Application and Theory of Petri Nets*, pages 95–112, 1988.
- [8] A. Valmari. A Stubborn Attack on State Explosion. In *Proceedings of Computer-Aided Verification '90*, volume 531 of *Lecture Notes in Computer Science*, pages 156–165. Springer-Verlag, 1990.
- [9] A. Valmari. Stubborn Sets for Reduced State Space Generation. In G. Rozenberg, editor, *Advances in Petri Nets '90*, volume 483 of *Lecture Notes in Computer Science*, pages 491–515. Springer-Verlag, 1990.
- [10] A. Valmari. The State Explosion Problem. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 429–528. Springer-Verlag, 1998.

A Compositional Model of Time Petri Nets

o n

Department of Computing Science, University of Newcastle
Newcastle upon Tyne NE1 7RU, U.K.

Abstract. This paper presents two related algebras which can be used to specify and analyse concurrent systems with explicit timing information. The first algebra is based on process expressions, called t-expressions, and a system of SOS rules providing their operational semantics. The second algebra is based on a class of time Petri nets, called ct-boxes, and their transition firing rule. The two algebras are related through a mapping which, for a t-expression, returns a corresponding ct-box with behaviourally equivalent transition system. The resulting model, called the Time Petri Box Calculus (tPBC), extends the existing approach of the Petri Box Calculus (PBC).

Keywords: Net-based algebraic calculi; time Petri nets; relationships between net theory and other approaches; process algebras; box algebra; SOS semantics.

1 Introduction

o n om on n om n m
m o n o o on n o ommon o mm n
on n on o m n o n o o
n on o m n on m n o
n o n n o n o n o
n on n n m o
n m n o n m n o on n
m o o n n n m n
m o o n o n composition-
ality n explicit asynchrony n n o on o m n
o n o
n mo m o o n n o
n o n n o o
n m o o no o on o m n on o
on m o on n m
n o n n o n mo o
on o on n m n n m n n o m on mo
n on o o n n n
m o n o n o on on n
on n o on o m

n o n on m n n
 n on o o m o n n
n n *earliest firing time* n *latest firing time*
o o o n on mom n om n
n m o o om o on
n m n o on n o on n o
no on m n m n no on n o
on m n o on n n m o
on n o o on n m o o
m
 o n o o on n n o on
n on n o o on m n on o
om o on n mo on m n n on n
m n o on n on n n o
on n no on m n o on
o o m m
on o n o n n no on o n
n m n on om o on m n o n

2 Syntax

n on n o o on o o

Static Expressions

om n on o o n on om
 n o o n n on n m
m o mo on n o o
m o communication on \mathcal{A} n $\hat{=}$ o
 \mathcal{A} conjugate $\hat{=}$ \mathcal{A} $\hat{=}$ n $\hat{=}$
n on silent o n n on \mathcal{A} m
synchronisation o o on omm n on on o n
on
o o n n o *static time box expressions* o
on

	sy \mathcal{A}		rs \mathcal{A}		\square		\parallel	
	sy \mathcal{A}		rs \mathcal{A}		\square			

on n n mo on n om n
n o on n on \mathcal{A}
on non n n n non n n
o n no *earliest* n *latest* on m o
m om mom n om n

n o n n o *ex-exclusive* on
o om o on n n no m n om
n on n m m n o n n n n n
n on om on o n n mo n on
o m o o o Ω o n n
on o on n
m o on m n mo n n m on
n n

Dynamic Expressions

n o n m on n
n m nno on o o n n nno on
non n n n o on n o *age* o n o o
n o m $-^3$ n on n n
o m n n $-_{23}$ n om o on om on n
m n m n o
dynamic t-expressions n o no
on no n on n **N**
— | — | sy A | rs A | \square | \square | || | K
K — | — | K sy A | K rs A | K \square | \square K | | |
| K |
o K n o n m on
o n m o oo m m n
on $-^6$ n n on n m n n o m on on o
n m n n n n o o
m on n o $-^6$ n on m o no
m on n on $-^6$ rs n nno
n o o n no o on
n n om m m mo m n
on

3 Operational Semantics of t-Expressions

n on n n o on m n o n m on
o o n o m n o n
n n on on on n n n o
o o on m n n o

3.1 Structural Equivalence

n on on on m o mo
n m n o on n n on o m —

o m o n m n om on n m n n
n o m n on om on n n n
on o n on n o n
n o n o o fl m mno on o o n
n o m n on on n m on
n

$\overline{E\ F}$	$\overline{E}\ \overline{F}$	$\underline{E}\ \underline{F}$, $\underline{E\ F}_{\min\{\ \prime\}}$
$\overline{E\ \Box\ F}$	$\overline{E}\ \Box\ \overline{F}$	$\underline{E}\ \Box\ \underline{F}$ $\underline{E\ \Box\ F}$
$\overline{E\ \Box\ F}$	$\overline{E\ \Box\ F}$	$\underline{E\ \Box\ F}$ $\underline{E\ \Box\ F}$
$\overline{E\ rs\ A}$	$\overline{E}\ rs\ A$	$\underline{E}\ rs\ A$ $\underline{E\ rs\ A}$
$\overline{E\ sy\ A}$	$\overline{E}\ sy\ A$	$\underline{E}\ sy\ A$ $\underline{E\ sy\ A}$
$\overline{E;F}$	$\overline{E}\ ;F$	$\underline{E;\underline{F}}$ $\underline{E;F}$
$\underline{E}\ ;F$	$\underline{E;\overline{F}}$	$[D\ \underline{E}\ F]$ $[D\ E\ \overline{F}\]$
$[\underline{D}\ E\ F]$	$[\overline{D}\ E\ F]$	$[D\ E\ \underline{F}\]$ $[\underline{D\ E\ F}]$
$[\underline{D}\ E\ F]$	$[D\ \overline{E}\ F]$	$[D\ \overline{E}\ F]$ $[D\ \underline{E}\ F]$

Table 1. o n

o no n o $\overline{\hspace{1cm}}$ \parallel $\overline{\hspace{1cm}}$ ' n on
n n om n m on on o
o m n o o o n on n n
on n m n o on on
 $\Box\ \overline{\hspace{1cm}}^1\parallel\overline{\hspace{1cm}}^0$
n o on n n n n o n o o o on n
mo o o on m n n o o o m o
o
 $\overline{\hspace{1cm}}\Box\parallel\gamma$
o om mm m n n o o n
o
 $\Box\ \overline{\hspace{1cm}}\parallel\overline{\hspace{1cm}}$
o o o on n m n o n m n
o o on o o m m om o n o

$$\begin{array}{c}
 \text{om} \quad \text{o} \quad \text{n} \quad \text{o} \quad \text{o} \quad \text{n} \quad \text{no} \\
 \text{o} \quad \text{o} \quad \text{m} \quad \text{mo} \quad \text{om} \quad \text{n} \quad \text{n} \\
 \text{on} \quad \text{n} \quad \text{n} \quad \text{on} \quad \text{on} \quad \text{om} \quad \text{m} \\
 \text{m} \quad \text{om} \quad \text{on} \quad \text{on} \quad \text{o} \quad \text{om} \\
 \text{o} \quad \text{n} \quad \text{m} \quad \text{on} \quad \text{n} \quad \text{on} \quad \text{n} \quad \text{on}
 \end{array}
 \quad \text{---} \parallel \text{---} '$$

Proposition 1. *Assuming that we treat the rules in table 1 as term rewriting rules, if γ is a time expression, then so is γ .*

3.2 SOS Rules

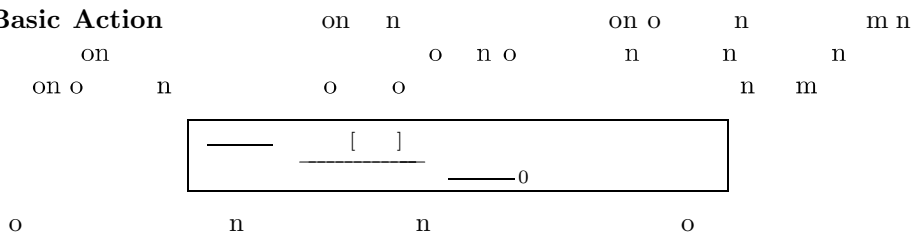
$$\begin{array}{c}
 \text{time} \quad \text{mo} \quad \text{o} \quad \text{n} \quad \text{o} \quad \text{o} \quad \text{on} \quad \text{m} \quad \text{n} \quad \text{mo} \quad \text{n} \quad \text{m} \quad \text{action} \quad \text{mo} \quad \text{n} \\
 \text{m} \quad \text{mo} \quad \text{o} \quad \text{m} \quad \text{---} \quad \text{m} \quad \text{o} \quad \text{n} \quad \text{n} \\
 \text{o} \quad \text{on} \quad \text{n} \quad \text{o} \quad \text{m} \\
 \text{n} \quad \text{on} \quad \text{mo} \quad \text{o} \quad \text{m} \quad \text{---} \quad \Gamma \quad \gamma_1 \quad \gamma \quad \text{n} \\
 \text{m} \quad \mathcal{A} \quad \gamma \quad \text{n} \quad \text{action occurrence} \quad \text{o} \quad \text{o} \quad \text{m} \\
 \text{n} \quad \text{non} \quad \text{n} \quad \text{n} \quad \text{on} \quad \text{o} \quad \text{n} \quad \text{on} \quad \text{non} \quad \text{n} \quad \text{n} \\
 \gamma \quad \text{legal} \quad \text{n} \quad \text{illegal} \quad \text{o} \quad \text{o} \quad \text{o} \quad \text{on} \quad \text{o} \quad \text{n} \\
 \text{n} \quad \text{on} \quad \text{o} \quad \gamma \quad \text{on} \quad \text{o} \quad \text{n} \quad \text{n} \quad \text{m} \quad \text{n} \quad \text{on} \quad \text{n} \\
 \text{mom} \quad \text{n} \quad \text{o} \quad \text{on} \quad \text{o} \quad \text{n} \quad \text{on} \quad \text{on} \quad \text{on} \quad \text{o} \\
 \text{n} \quad \text{o} \quad \text{no} \quad \text{n} \quad \text{n} \quad \text{on} \quad \text{o} \quad \text{o} \quad \text{n} \\
 \text{om} \quad \text{m} \quad \text{m} \quad \text{n} \quad \text{n} \quad \text{n} \quad \text{n} \quad \text{n} \\
 \text{on} \quad \text{n} \quad \text{on} \quad \text{o} \quad \text{n} \quad \text{n} \quad \text{n} \quad \text{on} \\
 \text{nno} \quad \text{o} \quad \text{on} \quad \text{o} \quad \text{n} \quad \text{o} \quad \text{n} \\
 \text{n} \quad \text{on} \quad \text{o} \quad \text{n} \quad \text{on} \quad \text{o} \quad \text{om} \quad \text{o} \quad \text{on} \quad \text{o} \quad \text{n} \\
 \text{m} \quad \text{m} \quad \text{n} \quad \text{m} \quad \text{n} \quad \text{o} \quad \text{on} \quad \text{o} \quad \text{n} \quad \text{o} \\
 \text{n} \quad \text{m} \quad \text{no} \quad \text{m} \quad \text{n} \quad \text{on} \quad \text{n} \quad \text{n} \quad \text{o} \quad \text{n} \\
 \text{on} \quad \text{o} \quad \text{o} \quad \text{o} \quad \text{n} \quad \text{on} \quad \text{on} \quad \text{n} \\
 \text{o} \quad \text{n} \quad \text{on} \quad \text{o} \quad \text{o} \quad \text{mo} \quad \text{o} \quad \text{o} \\
 \text{on} \quad \text{m} \quad \text{n} \quad \text{o} \quad \text{n} \quad \text{m} \quad \text{on}
 \end{array}$$

Empty Moves

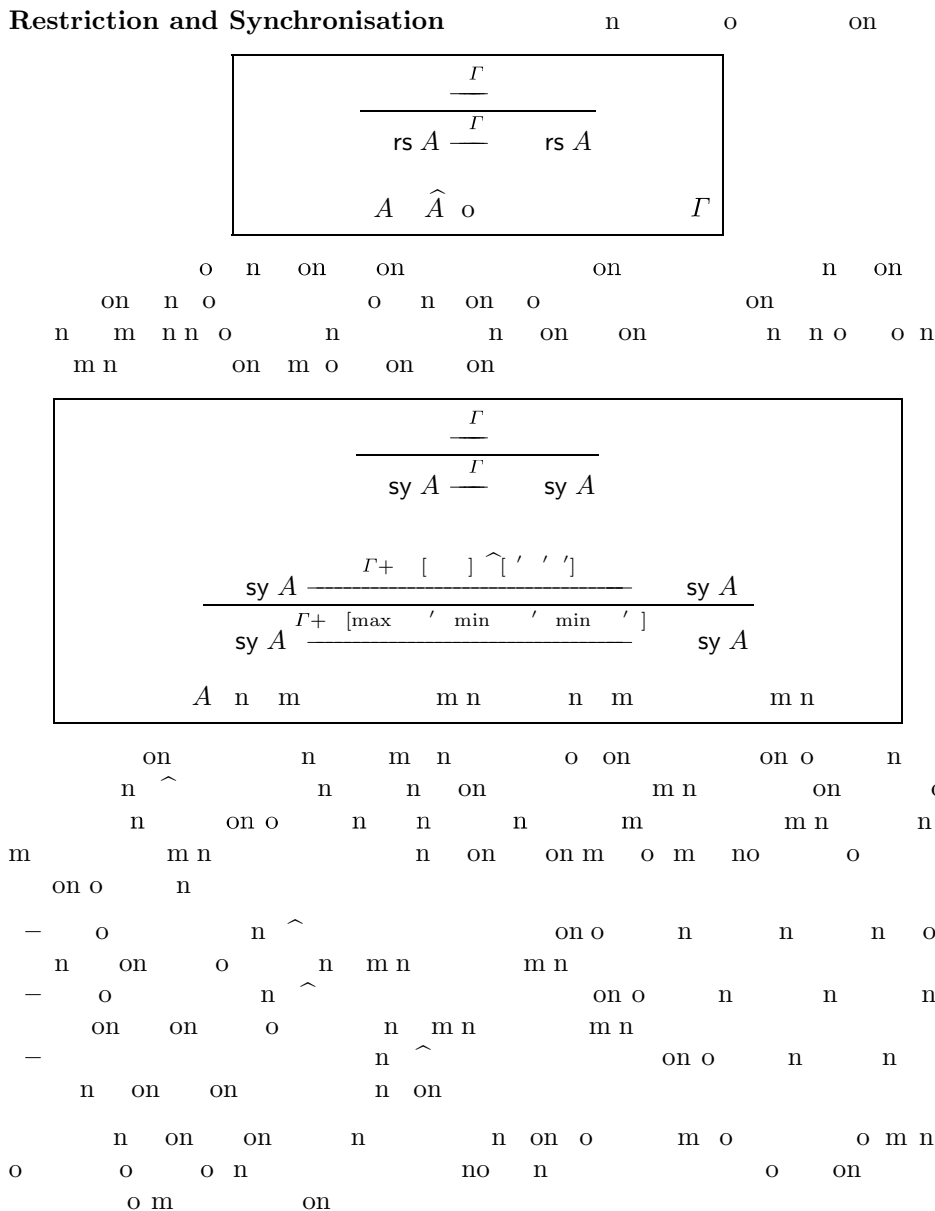
$$\text{o} \quad \text{o} \quad \text{n} \quad \text{m} \quad \text{on} \quad \text{mo}$$

$ \frac{}{} $	$ \frac{}{} $	$ \frac{}{} $
-----------------	-----------------	-----------------

Basic Action



Restriction and Synchronisation



Other tPBC Operators

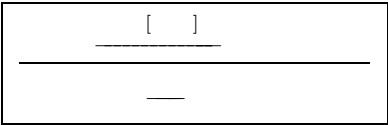
no n n o
m n n o o n om n
n n

$\frac{G \xrightarrow{\Gamma} G', H \xrightarrow{\Gamma'} H'}{G \ H \xrightarrow{\Gamma+\Gamma'} G' \ H'}$	$\frac{G \xrightarrow{\Gamma} H}{G \Box E \xrightarrow{\Gamma} H \Box E}$
$\frac{G \xrightarrow{\Gamma} H}{E \Box G \xrightarrow{\Gamma} E \Box H}$	$\frac{G \xrightarrow{\Gamma} H}{G; E \xrightarrow{\Gamma} H; E}$
$\frac{G \xrightarrow{\Gamma} H}{E; G \xrightarrow{\Gamma} E; H}$	$\frac{G \xrightarrow{\Gamma} H}{[G \ E \ F] \xrightarrow{\Gamma} [H \ E \ F]}$
$\frac{G \xrightarrow{\Gamma} H}{[E \ G \ F] \xrightarrow{\Gamma} [E \ H \ F]}$	$\frac{G \xrightarrow{\Gamma} H}{[E \ F \ G] \xrightarrow{\Gamma} [E \ F \ H]}$

Table 2. o o on m n

Time Moves

n m



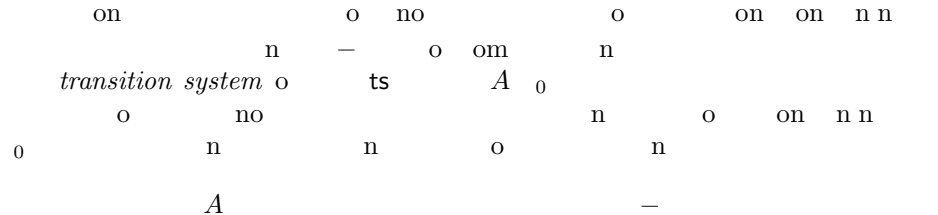
m nno on no o n n o
o o n m n n n
m n n m mo no n m o n o
on mo o m mo n on o mo
o n on nno o o on
n on n m o n o m
n n o o on m n o no o
o n m on o o on

Reachable t-Expressions

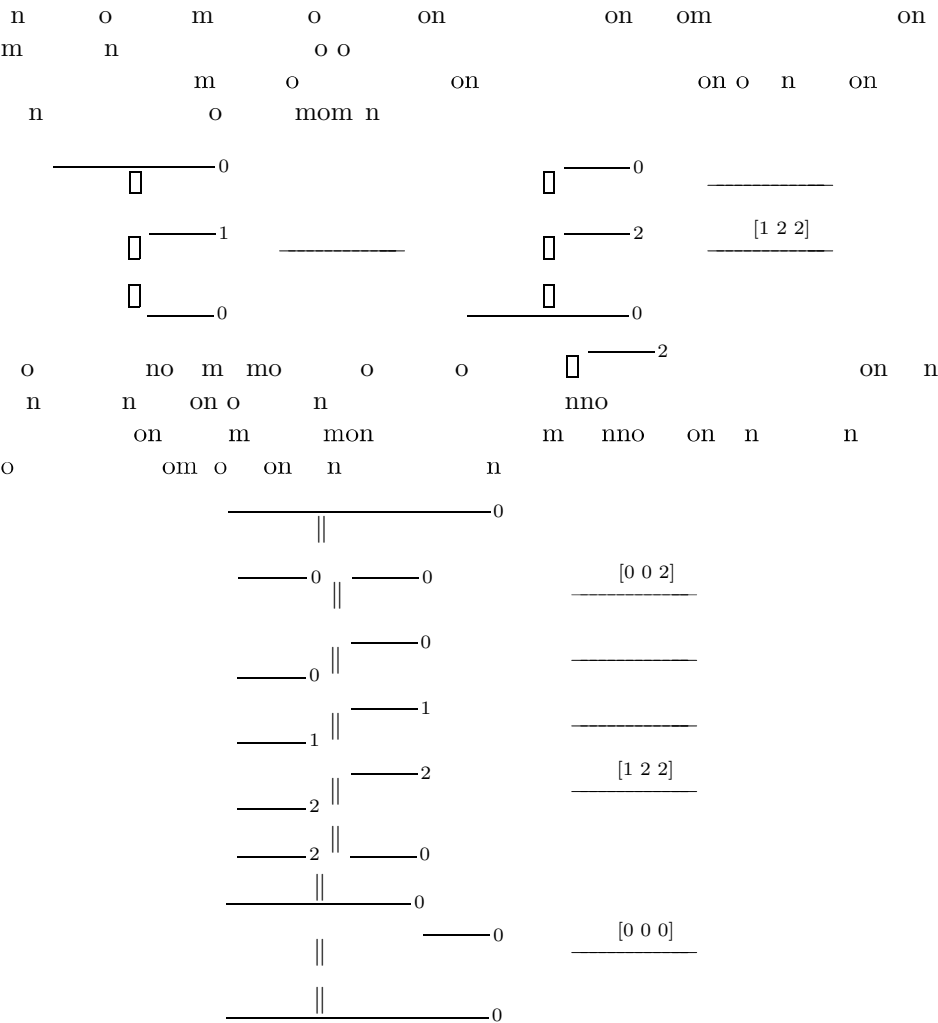
m n n o on
n o o on m n
om on o m n o o
n m on *reachable* on
n om ⁻⁰ n o on m n n n
on o o on n n
on o n

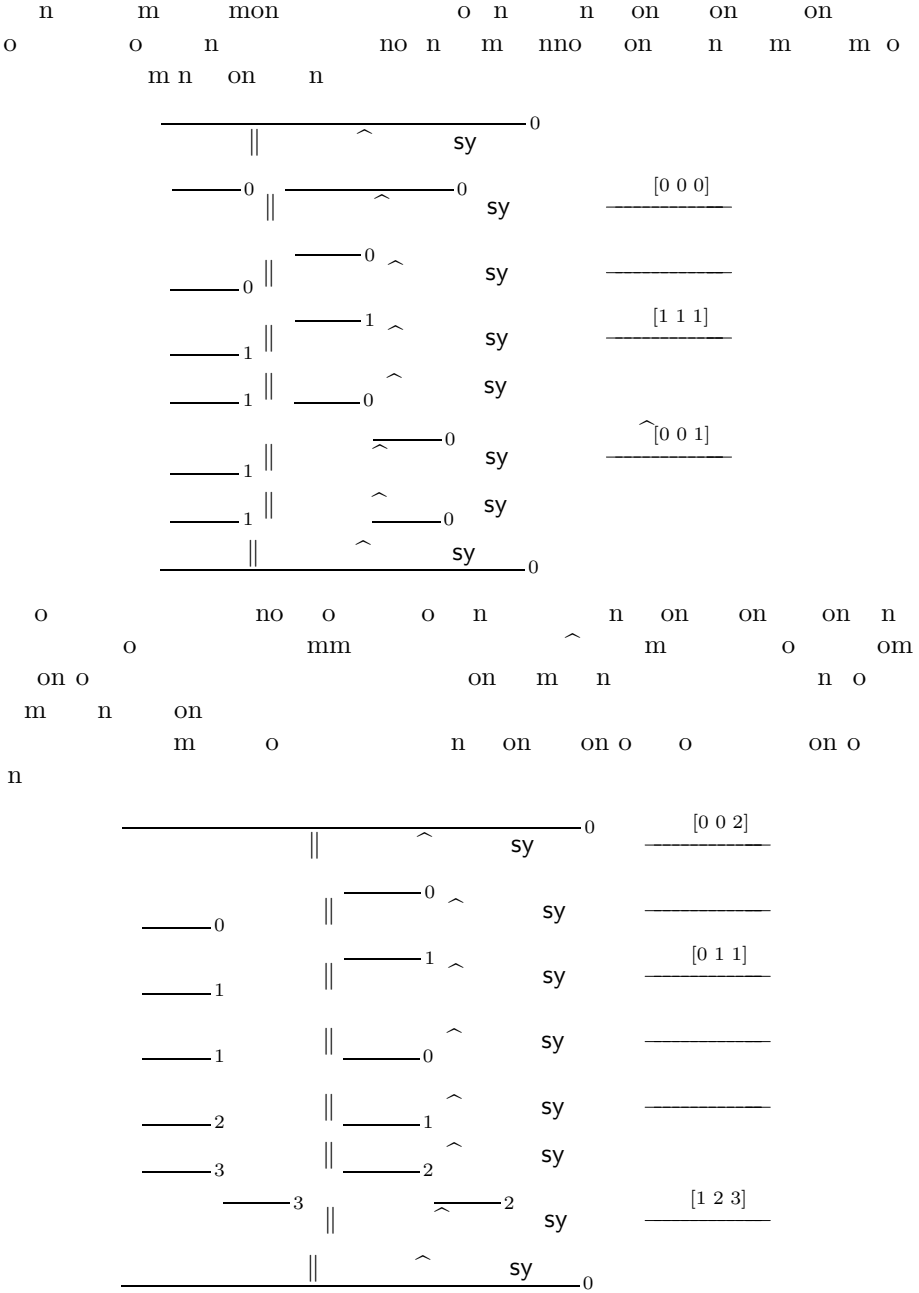
Proposition 2. *Let t be a reachable t-expression and $\Gamma \vdash t$. Then, every action occurrence in Γ is legal.*

Transition Systems of Reachable t-Expressions

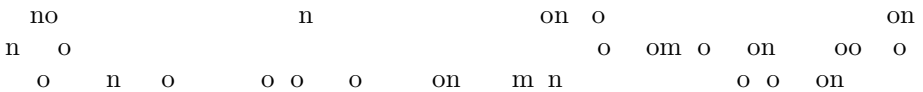


3.3 Examples





3.4 Operational Semantics of Subexpression



o n on n n o on
 n n o m n n o n on m n o on
 on n o oo o on o n
 n o m o o on no o on o
 on n m mo n o
 on o o n on — 0 rs n
 n — — 1 rs
 o o oo o on om o n o o
 on — 0 n m mo mo n o n
 — 0 — 1
 o no o m n o o o
 m on n n o on o on o on
 n n o o n omm n on on
 o n n no
 {a} — 1
 o n m mo on o on n
 on m o o n oo n
 on m no no n o n n on on o
 n m o n o — 0 || ^ — 0 n on n
 on o n on
 rs — — 1 || ^ — 1 rs
 on n n o m n m mo
 {a} — 1 || ^ — 1
 o o n rs ^
 on on m on n n
 on sy rs n n m n o on
 m mo o o
 — — 1 || ^ — 1 sy rs
 o m o n n n on o n
 o o n n o n n
 o m mo o on n o n

o m no n o o n o o
on m n n n n on on
nno o nno n n m mo
no on on o o
n on o n n

$$\frac{\emptyset, \{a\}}{\quad} \text{---} 1 \parallel \text{---} 1$$

o n n m m mo n on n no on
n on n

$$\frac{\{a\}, \{a\}}{\quad} \text{---} 1 \parallel \text{---} 1$$

no m mo
no o m o o on o o o om
m n on on $A B$ \mathcal{A} n o o n m
on no on o n n on
sy A rs B n n on n n n o
on n m on o o B
o n on on n A o on o
o on o on n om n n A om on
on o \wedge n n n sy A no on
sy/re-context o n n o n on n o n
internal m n

$$\frac{\text{sy } A \text{ rs } B \text{ --- } \text{sy } A \text{ rs } B}{\text{---} \frac{A, B}{\quad}}$$

o n on on n n on
n n o n m on
 $A B$ on o n o o m mo n
m mo o m o o on
n o on o on n om o on n
o o o

4 An Algebra of Time Boxes

n on n o o n o n m
n o n n om
n m nno on n on m n om o m
o n o o o on m o o on n o o n o
on

4.1 Time Boxes

time box o o *eft lft*
o n m o n on n \mathcal{A}
n n n *eft lft* N m n n o
n on *earliest firing time* n *latest firing time*
m *eft lft* o o n no
on n m no o om o n *static* m n
o m n *dynamic* m n o non m o on
n o m n o o n o
o
o on no on m o o n on skip
n redo n no sr skip redo m skip
n redo n o o n on n o o
n n n n n o m n
o o no on n n o
n o n o n on

Clock Vectors *clock vector* o m n sr N —
n skip n redo no o n n m n o

o o *semi-consistent* $^{-1}$ N enabled n m on n
o o *consistent* *lft* o
o N o o n o
sr o o n o o n o sr

 $\left\{ \begin{array}{l} \text{enabled } \overline{} \\ - \quad o \end{array} \right.$ $\left\{ \begin{array}{l} \text{enabled } \underline{} \\ - \quad o \end{array} \right.$

o o n on — no — o
on on n o o o o
clocked t-box o o o n
m on n o o o *consistently clocked t-box* o o
o o o o on n
n n on n n o n om
on n o o on n o o n o
o on n o m n om o n m
o o no on n o o o om o n on
mo o on o on o n o on m
n on on m on n on o
oo o m

Execution Semantics of ct-Boxes o n o o o
o n m m n on

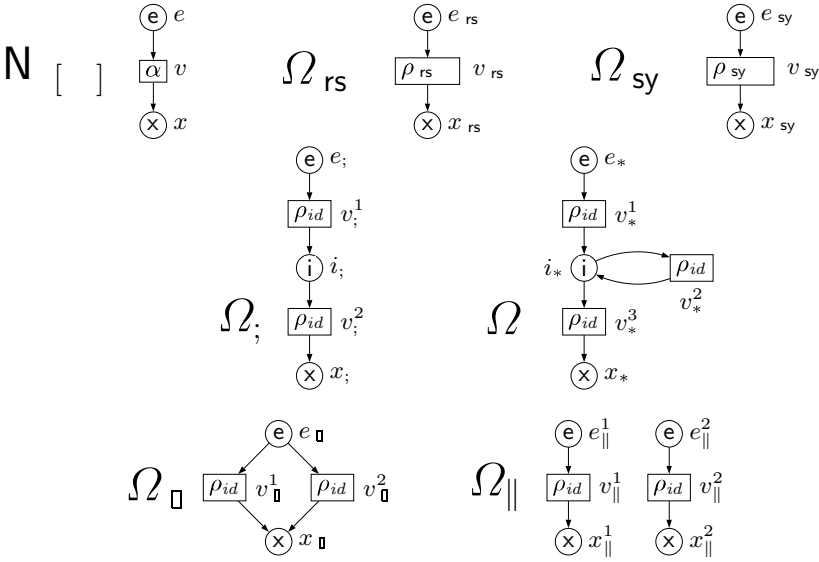


Fig. 1. o N [] eft n lft n o o

no o on n n om o on o o n o
n n m o o om o n o mo n
o om o n n n o
- o o n A A n
1 sy A 1 sy A - 1 rs A 1 rs A -
1 □ 2 1 □ 2 - 1 || 2 1 || 2 -
1 2 1 2 - 1 2 3 1 2 3 -
n om o n n m o on n o n o on
o o m n om n m o o n
n o no o n o n n
m n n o o o skip n redo

Initial and Final State - o n N n
- - n - -
o - o n on lft o n on n
n n m n o n o on n
no n on n n m n o n on o
o o on o

n m o n

A \mathcal{A} n on $on\ o$ A n
 $sy\ A$ $sy\ A$
 $skip$ $skip$ n $redo$ $redo$ $o\ o$ o n on
 sy \triangleleft 1 n $sy\ A$ $m\ n$ 1 o
 n on $on\ o$ n on 1 n o o
 m $o\ m\ o$ on n
 $n\ 1$ $n\ 2$ on $omm\ n$ on n
 $on\ o$ A n
 $rs\ A$ $rs\ A$
 $skip$ $skip$ n $redo$ $redo$ $o\ o$ o n on
 rs \triangleleft n $rs\ A$ \hat{o} on n on
 n no $on\ n$ $o\ A$ \hat{A}

O

o 1 n n m n 2 o om o on o 1 n 2
n
1 2 1 2
skip 1 skip n redo 1 redo o o o n on
◁ n 1 2 no on n on n
{ 1 skip enabled 1 2
o
o m 1 n n n m n o m n
n n on n n n m n o 2 o o
om o on 2 1 n m
n om o on o 1 n 2 n
1 2 1 2
skip 1 skip n redo 2 redo o o o n on
; ◁ n 1 2 no on n on n
{ 1 redo enabled 1 2
o
1 n n m n o m n n n on
n n n m n o 2 o o n
om o on o 2 n 1 n
2 1 2 1
skip 2 skip n redo 1 redo o o o n on
; ◁ n 2 1 no on n on n 2
n n 1
{ 1
2

Parallel Composition

$$\begin{array}{c}
\text{om } o \quad \text{on } o \quad 1 \quad n \quad 2 \quad n \quad o \quad o \quad n \quad m \quad o \\
1 \parallel 2 \quad 1 \parallel 2 \\
\text{skip} \quad m \quad n \quad 1 \quad \text{skip} \quad 2 \quad \text{skip} \quad n \quad \text{redo} \quad m \quad n \quad 1 \quad \text{redo} \quad 2 \quad \text{redo} \\
o \quad o \quad o \quad n \quad \text{on} \quad n \quad \text{on} \quad n \quad \triangleleft \quad n \quad 1 \parallel 2 \quad o \\
\text{on} \quad n \quad \text{on} \quad n
\end{array}$$

Iteration

$$\begin{array}{c}
n \quad m \quad o \quad o \quad o \quad \text{on } o \quad 1 \quad 2 \quad n \quad 3 \quad n \quad m \quad n \\
1 \quad 2 \quad 3 \quad 1 \quad 2 \quad 3 \\
\text{skip} \quad 1 \quad \text{skip} \quad n \quad \text{redo} \quad 3 \quad \text{redo} \quad o \quad o \quad o \quad n \quad \text{on} \\
\triangleleft \quad n \quad 1 \quad 2 \quad 3 \quad \text{no} \quad \text{on} \quad n \quad \text{on} \quad n \\
\left\{ \begin{array}{l} \text{enabled} \quad \text{enabled} \quad 1 \quad 2 \quad 3 \\ o \end{array} \right. \\
m \quad 1 \quad \text{redo} \quad 2 \quad \text{skip} \quad 2 \quad \text{redo} \quad 3 \quad \text{skip} \quad o \quad n \quad n \quad o \\
2 \quad n \quad m \quad o \quad n \quad n \quad n \quad m \quad n \quad o \quad m \quad n \\
o \quad o \quad m \quad m \quad n \quad n \quad \text{on} \quad n \quad n \\
n \quad m \quad n \quad o \quad 2 \quad n \quad 3 \quad o \quad o
\end{array}$$

5 Denotational Semantics of t-Expressions

$$\begin{array}{c}
n \quad \text{on} \quad n \quad o \quad \text{no} \quad \text{on} \quad m \quad n \quad o \quad \text{on} \\
m \quad n \quad m \quad n \quad \text{ctbox} \quad \text{om} \quad \text{on} \quad o \quad o \quad n \quad \text{om} \quad o \quad \text{on} \\
\text{ctbox} \quad N \quad [\quad] \quad - \quad N \quad [\quad] \quad o \quad n \quad n \quad o \\
o \quad n \quad n \quad m \quad \text{on} \quad n \\
\text{ctbox} \quad \text{ctbox} \quad \text{ctbox} \quad \text{ctbox} \quad \text{ctbox} \quad \text{ctbox} \quad \text{ctbox} \quad \text{ctbox} \\
\text{ctbox} \quad \text{sy } A \quad \text{ctbox} \quad \text{sy } A \quad \text{ctbox} \quad \text{rs } A \quad \text{ctbox} \quad \text{rs } A \\
\text{ctbox} \quad \square \quad \text{ctbox} \quad \square \text{ctbox} \quad \text{ctbox} \quad \parallel \quad \text{ctbox} \quad \parallel \text{ctbox} \\
\text{ctbox} \quad \text{ctbox} \quad \text{ctbox} \quad \text{ctbox}
\end{array}$$

Proposition 3. *The mapping ctbox always returns a ct-box and, for every reachable t -expression, it returns a cct-box. Moreover, if t and t' are dynamic t -expressions, then $\text{ctbox } t = \text{ctbox } t'$.*

5.1 Consistency between Denotational and Operational Semantics

no o m n o o m n
o on n on m n om
n o n o o o on m n om
on n m o o

Theorem 1. For every reachable t -expression ,

$ctbox$ is a node of ts

is a strong bisimulation between the transition systems ts and $ts_{ctbox()}$. The latter means that $ctbox$, and if then the following hold:

- If – then there is such that – and .
- If – then there is such that – and .

In other words, ts and $ts_{ctbox()}$ are strongly equivalent transition systems.

o n o n n on
m o o on n on n o omo o
o n n on m n o m no n n omo
o on on

rs

o n o on m n

0 — 1 — 2 —

n o o o $ctbox$ $ctbox$ o n
on o n o n m o –

–

n o n m mo n non n on
n n n o m n o
n o n no n n

Remarks on the Proof of Theorem 1

oo o o m o o
o o n n o n on on o
on o n mo n o n
o on o on n o n o o n no on
o on o n o o m o
o o omm n on on $A B$ \mathcal{A} n om n
o sy $A rs B$ o o o

$\xrightarrow{A,B}$ n on o sy $A rs B$ — sy $A rs B$

no n o on
 ctbox o o on n n o n
 n o n on m on on n
 m mo n n n o n on m o n
 ctbox on n n o m o o n
 on on o n no o n

5.2 An Example Motivating Illegal Actions

no n o on o n n o n
 o on n n on n n on n o
 o on n on || ^ sy
 om on n o on n o o n n
 o 0 nno n on on on on n
 n m n o on n o n o on n o
rs 0 n o o o o o o
 m n on n o on m n o o om
 o o n m n o n on n on on
 o n

|| ^ sy rs 0
0 || ^ sy rs
1 || 1 ^ sy rs [1 1 1]
1 || ^ sy rs
1 || 0 ^ sy rs
1 || ^ 0 sy rs

no o n o n on on n on o
 o n om n on o n ^ n
 on on o n

1 || 0 [0 1 0] ^ [0 0 1] 0 || ^ 0
 on n on on

1 || 0 sy [0 0 1] 0 || ^ 0 sy
 o o om o n on n o on

1 || 0 sy rs [0 0 1]
0 || ^ sy rs
 || ^ sy rs 0

on on on o no o m n no
 on o n o o 1

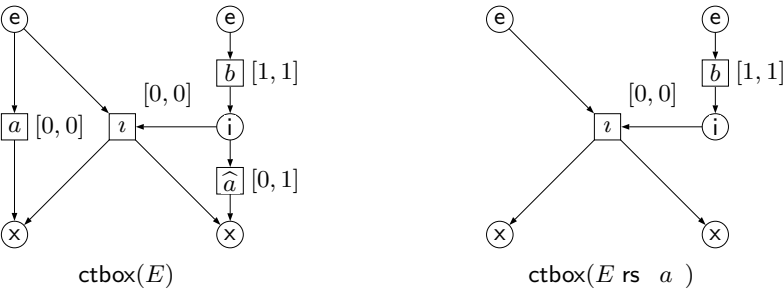


Fig. 2.

5.3 Time Divergence

on time guarded

— sy A o rs A n m

— \square n o n m

— o || o n o m

n on non-time-divergent o on

m

6 Concluding Remarks

n o mo o o on n m n o o

o on m n n o m on n o n

m n n o m mo n n o

n mo n n n n o o mo om

on n o n on n

o n n n om o o on m n

o no n n o on n mo n n

o o o o o o n o o

Acknowledgements

o o n n o o o on m

o n n o n o o

omm n on o on o o
 n n o n o m n o n on o
 o m

References

1. J. Baeten and W. P. Weijland: *Process Algebra*. Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press (1990).
2. B. Berthomieu and M. Diaz: Modelling and verification of Time Dependent Systems Using Time Petri Nets. *IEEE Trans. on Soft. Eng.* 17 (1991) 259-273.
3. E. Best, R. Devillers and J. Esparza: General Refinement and Recursion Operators for the Petri Box Calculus. Proc. of *STACS'93*, P. Enjalbert, A. Finkel and K. W. Wagner (Eds.). Springer-Verlag, Lecture Notes in Computer Science 665 (1993) 130-140.
4. E. Best, R. Devillers and J. Hall: The Petri Box Calculus: a New Causal Algebra with Multilabel Communication. In: *Advances in Petri Nets 1992*, G. Rozenberg (Ed.). Springer-Verlag, Lecture Notes in Computer Science 609 (1992) 21-69.
5. E. Best, R. Devillers and M. Koutny: Petri Nets, Process Algebras and Concurrent Programming Languages. In: *Advances in Petri Nets. Lectures on Petri Nets II: Applications*, W. Reisig and G. Rozenberg (Eds.). Springer-Verlag, Lecture Notes in Computer Science 1492 (1998) 1-84.
6. E. Best, R. Devillers and M. Koutny: The Box Algebra - a Model of Nets and Process Expressions. Proc. of *ICATPN'99*, S. Donatelli and J. Kleijn (Eds.). Springer-Verlag, Lecture Notes in Computer Science 1639 (1999) 344-363.
7. E. Best, R. Devillers and M. Koutny: *Petri Net Algebra*. EATCS Monographs on Theoretical Computer Science, Springer-Verlag (to be published in 2000).
8. B. Bieber and H. Fleischhack: Model Checking of Time Petri Nets Based on Partial Order Semantics. Proc. of *CONCUR'99*, J. Baeten and S. Mauw (Eds.). Springer-Verlag, Lecture Notes in Computer Science 1664 (1999) 210-225.
9. H. Fleischhack: Computing a Complete Finite Prefix of a Time Petri Net. Draft paper (1999).
10. M. B. Hennessy: *Algebraic Theory of Processes*. The MIT Press (1988).
11. C. A. R. Hoare: *Communicating Sequential Processes*. Prentice Hall (1985).
12. R. Janicki and P. E. Lauer: *Specification and Analysis of Concurrent Systems - the COSY Approach*. EATCS Monographs on Theoretical Computer Science, Springer-Verlag (1992).
13. M. Koutny and E. Best: Fundamental Study: Operational and Denotational Semantics for the Box Algebra. *Theoretical Computer Science* 211 (1999) 1-83.
14. P. Merlin and D. Farber: Recoverability of Communication Protocols - Implication of a Theoretical Study. *IEEE Trans. on Soft. Comm.* 24 (1976) 1036-1043.
15. R. Milner: *Communication and Concurrency*. Prentice Hall (1989).
16. T. Murata: Petri Nets: Properties, Analysis and Applications. *Proc. of IEEE* 77 (1989) 541-580.
17. G. D. Plotkin: A Structural Approach to Operational Semantics. Technical Report FN-19, Computer Science Department, University of Aarhus (1981).
18. L. Popova: Time Petri Nets. *Journal of Information Processing and Cybernetics EIK* 27 (1991) 227-244.
19. W. Reisig: *Petri Nets. An Introduction*. EATCS Monographs on Theoretical Computer Science, Springer-Verlag (1985).

Composing Abstractions of Coloured Petri Nets

Charles Lakos

Computer Science Department

University of Adelaide

Adelaide, SA, 5005, Australia.

Charles.Lakos@adelaide.edu.au

Abstract: An earlier paper considered appropriate properties for abstract net components (or nodes) in the Coloured Petri Net formalism. This paper augments that earlier work in three main areas — it proposes general canonical forms for such node refinements, it identifies two other forms of refinement which will be used in concert with node refinement, and it considers the compositionality of these refinements. All of them maintain behavioural compatibility between refined and abstract nets, which is captured by the notion of a system morphism.

Keywords: Theory of High-Level Petri Nets, Abstraction, Refinement

1 Introduction

The abstraction (and refinement) of Petri Nets is not new and dates back to Carl Adam Petri himself (e.g. [16]). One approach has been to use abstraction to focus on the structural relationship between the abstract and refined nets in terms of net morphisms [1, 3, 4, 6, 17]. In the case of Free Choice nets, the structural compatibility has behavioural implications, namely the maintenance of traps and syphons. In the context of High-Level Petri Nets (such as CPNs [9] and Predicate-Transition Nets [8]), the approach adopted has typically been to use abstraction to aid the process of developing a Petri Net model, with the abstraction subsequently being discarded when the model is simulated or analysed [17]. Thus CPNs, as formalised by Jensen [9] and implemented in the Design/CPN tool [10], provide substitution transitions together with place fusion for building Hierarchical Coloured Petri Nets (HCPNs). While substitution transitions maintain structural compatibility, there is no concept of abstract behaviour, since the semantics of the construct are defined in terms of the underlying CPN — in fact, the inscriptions on the arcs incident on substitution transitions are simply ignored.

By contrast, we prefer a view of abstraction in CPNs which respects behavioural constraints (as in [15, 18]), so that the abstraction can contribute to the understanding of the net and also to its analysis. In discussing behaviour-respecting morphisms for Petri Nets, Winskel [18] raised the question of the appropriate form of morphisms for CPNs. This paper advocates three forms which respect the colour structures — *type*, *subnet* and *node* refinement. Each of these maintains behavioural compatibility between refined and abstract nets. For type refinement, this means that the refined type values can be projected onto the abstract type values; for subnet refinement (or extension), the refined behaviour can be restricted to yield the corresponding abstract behaviour; for node refinement, a subnet which is abstracted by a place should maintain the notion of an abstract marking, and a subnet which is abstracted by a transition should have the notion of abstract firing.

An earlier paper focussed on the notion of abstraction for Coloured Petri Nets (CPNs) [12], in the sense of mapping place (or transition) bordered subnets into a

single place (or transition). That paper sought to be as general as possible, while maintaining behavioural compatibility between the abstract and refined nets. Thus, if a place-bordered subnet were to be abstracted to a place, then there needed to be some notion of an abstract marking for the subnet, which would only be modified by interaction of the subnet with its environment (and not by internal actions of the subnet). Similarly, if a transition-bordered subnet were to be abstracted to a transition, then there ought to be some notion of abstract firing for the subnet, where a firing sequence of the subnet included the firing of border transitions with matching firing modes.

The generality of the above proposals led to difficulties in proving compositionality results [13] — it was necessary to constrain the previous definitions in order to prove that the composition of two node refinements was also a node refinement. Thus, a place refinement was required to be resettable, i.e. a return to the same abstract marking could be matched by a return to the same refined marking (of the subnet). Similarly, a transition refinement should be able to return to the initial marking (of the subnet) after each abstract firing.

The above raised a dilemma as to whether the more general or the more restricted definitions were appropriate. This paper resolves the dilemma by asserting paradoxically that both are appropriate. This is achieved by recognising that three forms of refinement — type, subnet, and node refinement — are normally used in concert. This means that we can propose canonical bases for node refinements which satisfy the more constrained definitions, but in conjunction with the other forms of refinement we can achieve the generality of the original formulation.

The current paper is intended to be self-contained, but some aspects from the earlier paper are reiterated in abbreviated form. It is therefore desirable that the reader has access to the earlier paper. An informal introduction together with a brief motivating example is presented in §2. The definitions of CPNs and their refinement are found in §3. In §4, we prove the generality of the canonical node refinements. Finally, further comments on the use of these refinements and the conclusions are given in §5.

2 Example

This section presents a simple motivating example for the refinement proposals. The example is that of a library loans system and is adapted from an example found in [5].

A book in the library will have some associated information, such as the author(s), title, publisher, etc. In a CPN, each book can be represented by a token, and this information can be captured in a token type called *Book*. Each book will be in one of several states such as available, on loan, and overdue. These states can be represented by places in the Petri Net, with the presence of a book token in a place indicating that the book is currently in that state. Thus a CPN for the library books could be as shown in fig 2.1.

In this case, we have not shown all the details of the net. Thus, the colour for the token type is not indicated in full, nor is the relation between the transition firing modes and the variables inscribing the arcs. Clearly, however, the firing mode for transition *Borrow* will include sufficient information to identify the book *b* and the user *u*.

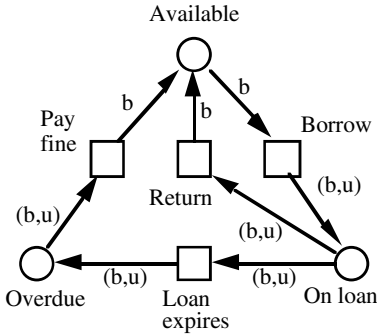


Fig 2.1: CPN Books — the lifecycle of library books

A borrower or user of the library will have some associated information, such as their name, contact details, classification of membership, etc. Again, each user can be represented by a token type called *User*, and the various states of the user can be indicated by places. In this case, it is convenient to have only one place to indicate the state of the user with transitions indicating the possible actions such as borrowing or returning a book and paying a fine. Thus, a CPN for the library users could be as shown in fig 2.2.

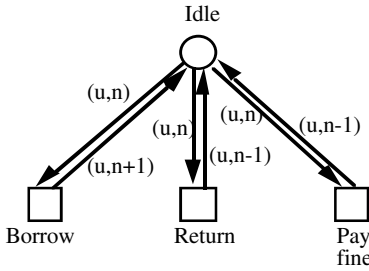


Fig 2.2: CPN Users — the lifecycle of library users

The two CPNs above could be combined into a composite system by fusing the similarly-named transitions in the two nets. Clearly, many more details could be specified, but the example above will be adequate for our purposes of illustrating the forms of refinement which we wish to support.

The first and simplest form of refinement, which we call *type refinement*, is to incorporate additional information in the net in the tokens and firing modes. However, each value of the refined type can be projected onto a value of the abstract type. For example, it may be desirable to introduce a further classification of books to vary the loan period. As far as the subnet for the books is concerned, this will simply involve extending the token type for the places, and extending the corresponding type for the various transition firing modes. The projection from refined to abstract type values is obvious. The above changes will affect the firing of the *Loan expires* transition, especially in the composite system. It is certainly the case, however, that if there is a behaviour of the refined system, then there would be a corresponding behaviour of the abstract system. This is the generic behavioural constraint which we require for acceptable refinements.

The second form of refinement, which we call *subnet refinement*, is to augment a subnet with additional places, transitions and arcs. (We also classify as subnet refinement

Colours:

$C(\text{Available}) = \text{Book}$

$C(\text{On loan}) = \text{Book} \times \text{User}$

$C(\text{Overdue}) = \text{Book} \times \text{User}$

$C(\text{Borrow}) = \text{Book} \times \text{User}$

$C(\text{Return}) = \text{Book} \times \text{User}$

$C(\text{Loan expires}) = \text{Book} \times \text{User}$

$C(\text{Pay fine}) = \text{Book} \times \text{User}$

Colours:

$C(\text{Idle}) = \text{User} \times \text{Limit}$

$C(\text{Borrow}) = \text{User} \times \text{Limit}$

$C(\text{Return}) = \text{User} \times \text{Limit}$

$C(\text{Pay fine}) = \text{User} \times \text{Limit}$

the extension of a token type or mode type to include extra values which are independent of previous processing. Here, these values of the extended type are not projected onto values of the abstract type but are ignored in the abstraction.) For example, it may be appropriate to cater for the reservation of books. In this case, we would add a place to hold the reservation status for each book, and the transition *Borrow* would only fire if there were a compatible reservation status on the book for the given user. (We use the value $(b,-)$ to indicate that no-one has a reservation for book b , and the value (b,u) to indicate that user u has a reservation on book b .) The modified part of the *Books* subnet is given in fig 2.3. Again we satisfy the constraint that if there is a refined behaviour, then there is a corresponding abstract behaviour (which ignores reservations), but not necessarily vice versa.

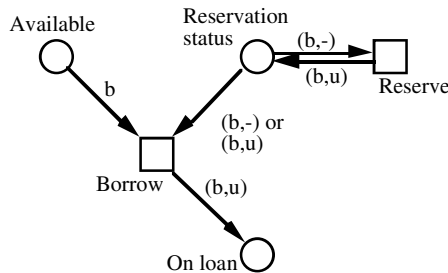


Fig 2.3: Subnet indicating the processing of book reservations

The third form of refinement, which we call *node refinement*, is to replace a place (or transition) by a place (or transition) bordered subnet. In this paper, we advocate the use of canonical forms of such refinements. The basis for a canonical place refinement is given in fig 2.4.

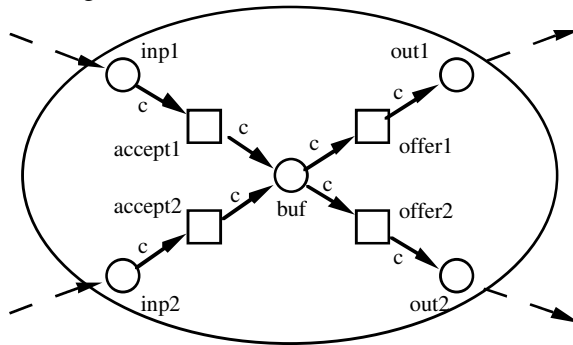


Fig 2.4: Canonical place refinement

It has separate input and output border places — in this case there are two of each. Each input (output) border place may have more than one incident input (output) arc from (to) the environment. Each input border place has an associated *accept* transition which will transfer tokens from the border place to an internal place, here called *buf*. Similarly, each output border place has an associated *offer* transition which will transfer tokens from place *buf* to the output border place. All the border places and the place *buf* have the same token type, which is also the mode type shared by the *accept* and *offer* transitions. None of these transitions constrains the flow of tokens, e.g. by using a guard. Clearly, the abstract marking of such a canonical place refinement is given by the sum of tokens in the border places and the internal place *buf*.

An arbitrary place refinement will be of the form of the basis of a canonical refinement (as above) augmented by subnet refinement which extends the *accept* and *offer* transitions. It is a contribution of this paper to observe how the combination of different refinements extends the generality of the basic constructs.

In our running example, such an incremental change might be the identification of the details of processing a book once it has been returned. In other words, the place *Available* might be replaced by a subnet which takes into account the delay in reshelving a book, the possibility of repairs, etc. The node refinement for this place is shown in fig 2.5.

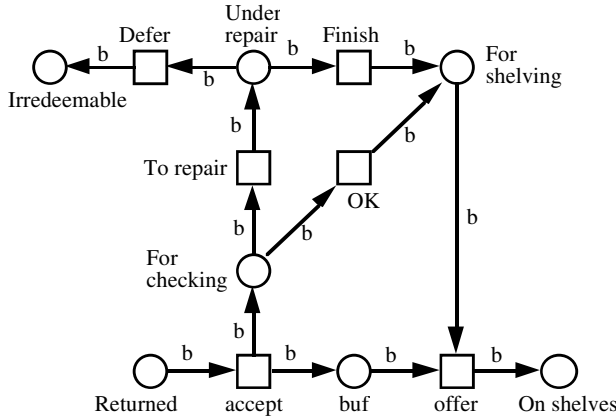


Fig 2.5: Subnet indicating the processing of returned books

Note that this has one input border place called *Returned* and one output border place called *On shelves*. The *accept* and *offer* transitions, together with the internal place *buf* constitute the basis of the canonical place refinement. Further activity is achieved by the subnet refinement which extends transitions *accept* and *offer*. This subnet retains the identity of the books, and hence it also determines the abstract marking of the subnet. Thus, the place *buf* is redundant, since its marking is equivalent to the sum of markings of places *For checking*, *Under repair*, *Irredeemable*, *For shelving*. (It is commonly the case that the place *buf* is redundant in such place refinements.) Further, this abstract marking is not modified by the various actions internal to the subnet. Clearly, a refined behaviour of the net will have a corresponding abstract behaviour, though the reverse will not necessarily be the case. For example, it may be that a book is so damaged that its return to the shelves would be indefinitely delayed, in which case further borrowing of that book would be disallowed.

For transition refinement, the canonical basis is given in fig 2.6. It has separate input and output border transitions — in this case there are two of each. Each input (output) border transition may have more than one incident input (output) arc from (to) the environment. Each input border transition has an associated place *recd*, which receives a token equal to the abstract firing mode, when the input border transition has fired with that mode. The transition *switch* can fire when all the input border transitions have fired (with the matching abstract firing mode), thereby completing the input phase. It removes the matching tokens from the *recd* places and puts corresponding tokens into all the *send* places. There is one such *send* place associated with each output border transition. Once such a token is available the

output border transition can fire (with the same abstract firing mode). Initially, all the *recd* and *send* places are empty.

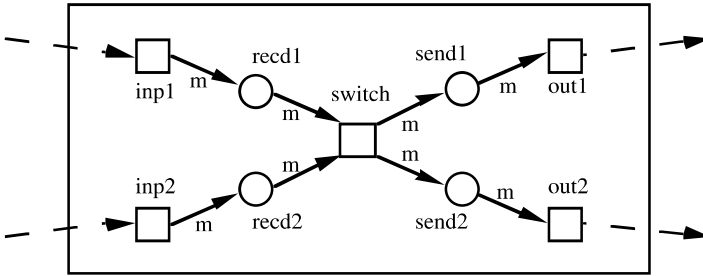


Fig 2.6: Canonical transition refinement

The abstract firing of the transition refinement commences with the firing of any of the input border transitions and is *completed* when all the matching output border transitions have fired and the *recd* and *send* places are again empty. Only such completed firing sequences will have corresponding abstract firing sequences. The canonical construction ensures that input border transitions fire before the corresponding output border transitions, ensuring the enabling of the corresponding abstract transition.

An arbitrary transition refinement will be of the form of a canonical refinement (as above) together with a subnet refinement which augments the border transitions. In our running example, we may wish to refine the *Borrow* transition to reflect the component activities of checking the student identity and processing the book. This might be achieved in the subnet of fig 2.7.

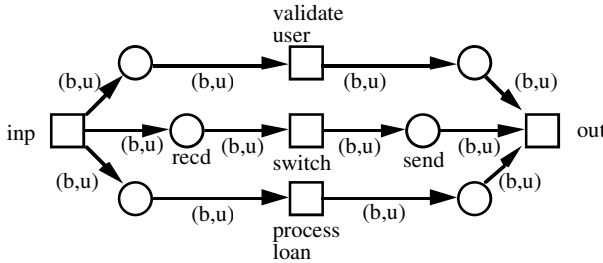


Fig 2.7: Refined Borrow transition

Note that the transition *validate user* may fail to fire (because the user is not acceptable). In this case, the abstract firing of this subnet will never complete, and hence such an incomplete refined activity will have no corresponding abstract activity.

Even though the above three forms of refinement can be clearly identified and analysed in isolation, they will commonly be used in concert in practical applications.

3 Formal Definitions

In this section we present the formal definitions of CPNs in §3.1, and the definition of CPN morphisms in §3.2. We identify the notion of system morphisms to capture the notion of refinement with behavioural compatibility, in contrast to the more traditional net morphisms which (only) specify structural constraints. The definition of our proposed refinements is given in §3.3. The definitions and their associated explanatory notes should be read together. The explanatory notes in §3.1 are minimal, since we assume that the reader has some familiarity with the definition of CPNs (e.g. [9]).

3.1 Formal definition of Coloured Petri Nets

We define Coloured Petri Nets in the context of a **universe of non-empty colour sets** Σ with an associated **partial order** $<: \subseteq \Sigma \times \Sigma$ which is derived from type compatibility in the theory of object-oriented languages [2]. $X <: Y$ means that the values of X can be used in contexts expecting values of Y , and typically it means that X has extra data components over Y . In this case, we assume the existence of a (polymorphic) projection function Π_Y from the values of X to those of Y (which do not appear in any proper subtype). For our purposes, we only make use of the fact that values of X are also values of Y .

Given a universe of colour sets Σ , we define $\Phi\Sigma = \{ X \rightarrow Y \mid X, Y \in \Sigma \}$, the functions over Σ , $\mu X = \{ X \rightarrow \mathbb{N} \}$ the multisets over X (where \mathbb{N} is the set of non-negative integers), and $\sigma X = \{ x_1 x_2 \dots x_n \mid x_i \in X \}$ the sequences over X . Where appropriate, multisets are elements of Σ , and thus mappings between multisets will occur in $\Phi\Sigma$ as required. Sum, difference and inequality are defined on multisets in the usual way, and a common notation for multisets [9] is adopted, e.g. $m = 3'b + 4'c$ is a multiset with $m(b) = 3$ and $m(c) = 4$, the only elements of m . We take the liberty of abbreviating $1'x$ as x . We usually name sequences with an asterisk suffix.

Definition 3.1: A **Coloured Petri Net** is a tuple $N = (P, T, A, C, E, \mathbb{M}, \mathbb{Y}, M_0)$ where:

- (a) P = set of places
- (b) T = set of transitions, with $P \cap T = \emptyset$
- (c) A = set of arcs, with $A \subseteq P \times T \cup T \times P$
- (d) C = colours of places and (modes) of transitions, i.e. $C: P \cup T \rightarrow \Sigma$
- (e) E = arc inscriptions with $E: A \rightarrow \Phi\Sigma$ where $E(p,t), E(t,p): C(t) \rightarrow \mu C(p)$
- (f) \mathbb{M} = set of markings, i.e. $\mathbb{M} = \mu\{(p,c) \mid p \in P, c \in C(p)\}$
- (g) \mathbb{Y} = set of steps, i.e. $\mathbb{Y} = \mu\{(t,c) \mid t \in T, c \in C(t)\}$
- (h) M_0 = initial marking, i.e. $M_0 \in \mathbb{M}$

Note: The above definition is, to all intents and purposes, equivalent to the common definition [9]. Note however, that there is at most one arc in each direction for any (place, transition) pair, and that we refer to the firing modes of transitions as colours and not as bindings. Thus, the effect of an arc is given by the arc inscription in conjunction with a particular mode, rather than the evaluation of an expression in a given binding. There is no guard function defined on transitions, but the same effect is achieved by limiting the associated set of firing modes. As usual, the markings of N are the multisets of (place, colour) pairs, while the steps are the multisets of (transition, colour) pairs. While these are derivative quantities, they are included in the tuple so that later it will be clear that morphisms map markings and steps to markings and steps respectively.

Definition 3.2: For CPN N , $x \in P \cup T$, $X \subseteq P \cup T$ we define:

- (a) **the inputs of x** , $\bullet x = \{ y \in P \cup T \mid (y,x) \in A \}$
- (b) **the outputs of x** , $x \bullet = \{ y \in P \cup T \mid (x,y) \in A \}$
- (c) **the border of X** , $\text{bd}(X) = \{ x \in X \mid \exists y \in P \cup T \setminus X: y \in \bullet x \cup x \bullet \}$
- (d) **the environment of X** , $\text{env}(X) = \{ y \in P \cup T \setminus X \mid \exists x \in X: y \in \bullet x \cup x \bullet \}$

Thus, the border of a set of nodes X , are those nodes connected to other nodes not in X , and those other nodes constitute the environment of X .

Definition 3.3: For CPN N , the **incremental effects** $E^-, E^+ : \mathbb{Y} \rightarrow \mathbb{M}$ of the occurrence of a step Y are given by:

$$(a) \quad E^-(Y) = \sum_{(t,m) \in Y} \sum_{(p,t) \in A} \{p\} \times E(p,t)(m)$$

$$(b) \quad E^+(Y) = \sum_{(t,m) \in Y} \sum_{(t,p) \in A} \{p\} \times E(t,p)(m)$$

Definition 3.4: For CPN N , a step $Y \in \mathbb{Y}$ is **enabled** in marking $M \in \mathbb{M}$, written $M [Y >$, if $M \geq E^-(Y)$. If step $Y \in \mathbb{Y}$ of CPN N is enabled in marking $M_1 \in \mathbb{M}$, it may **fire** leading to marking $M_2 \in \mathbb{M}$, written $M_1 [Y > M_2$, with $M_2 = M_1 - E^-(Y) + E^+(Y)$. A step sequence $Y^* = Y_1 Y_2 \dots Y_n \in \sigma \mathbb{Y}$ of CPN N is enabled in marking $M_1 \in \mathbb{M}$ and may occur, leading to marking $M_2 \in \mathbb{M}$, written $M_1 [Y^* > M_2$ if there exists intermediate markings $M_2', M_3', \dots M_n' \in \mathbb{M}$ such that $M_1 [Y_1 > M_2', M_i' [Y_i > M_{i+1}'$ for $i = 2, \dots n-1$, and $M_n' [Y_n > M_2$.

Definition 3.5: For CPN N , step $Y \in \mathbb{Y}$ is **realisable by** $Y^* \in \sigma \mathbb{Y}$ in marking $M_1 \in \mathbb{M}$ leading to marking $M_2 \in \mathbb{M}$ if $M_1 [Y^* > M_2$ and $\sum_{y \in Y^*} y = Y$.

Note: If a step Y is enabled in marking M , then it is realisable by Y .

Definition 3.6: For CPN N , the set of **reachable markings** $\mathbb{M}_R \subseteq \mathbb{M}$ is given by $\mathbb{M}_R = \{ M \in \mathbb{M} \mid \exists Y^* \in \sigma \mathbb{Y} : M_0 [Y^* > M \}$. The set of **enabled step sequences** $\mathbb{Y}^*_E \subseteq \sigma \mathbb{Y}$ is given by $\mathbb{Y}^*_E = \{ Y^* \in \sigma \mathbb{Y} \mid \exists M \in \mathbb{M} : M_0 [Y^* > M \}$.

3.2 Formal definition of CPN morphisms

This section defines morphisms between CPNs and distinguishes between net morphisms (which respect structure) and system morphisms (which respect behaviour). We allow morphisms to ignore components, but where components do have images they will be consistent, i.e. nodes will map to nodes and arcs to arcs. Further, given our interest in abstraction, we only consider morphisms which are surjective with respect to P' , T' and A' , i.e. refinements may add or modify components but will not delete them. Consequently, we can define the preimage of every node. (The fact that ϕ is surjective means that ϕ^{-1} is defined on P' and T' .)

Definition 3.7: Given a morphism $\phi: N \rightarrow N'$ and $x' \in P' \cup T'$, we define its **preimage** by $N_{x'} = \phi^{-1}(x')$ and write $N_{x'} = (P_{x'}, T_{x'}, A_{x'}, C_{x'}, E_{x'}, \mathbb{M}_{x'}, \mathbb{Y}_{x'}, M_{0x'})$.

Definition 3.8: A **net morphism** $\phi: N \rightarrow N'$ is a mapping from N to N' which is **structure-respecting**, namely:

- (a) ϕ is surjective with respect to P', T', A'
- (b) $\forall (x,y) \in A \cap P \times T \Rightarrow \phi(x) = \phi(y) \vee (\phi(x), \phi(y)) \in A' \cap (P' \times T')$
- (c) $\forall (x,y) \in A \cap T \times P \Rightarrow \phi(x) = \phi(y) \vee (\phi(x), \phi(y)) \in A' \cap (T' \times P')$

This is the common definition of net morphism (albeit with various additional constraints) which is primarily concerned with respecting the adjacency properties of the net [1, 3, 4, 6, 16, 17]. It does not constrain behaviour (except indirectly) — in fact, sets of markings and steps are not normally included. It also does not encompass restriction on places and transitions, i.e. where selected places and transitions (and their associated arcs) are ignored by the morphism. The compositionality of net morphisms is captured by the following proposition.

Proposition 3.9: Given two net morphisms $\phi_1: N \rightarrow N'$ and $\phi_2: N' \rightarrow N''$ their composition $\phi = \phi_2 \circ \phi_1: N \rightarrow N''$ is a net morphism.

Proof:

The composition of two surjective morphisms is surjective.

Consider $(x, y) \in A \cap P \times T$ and write $x' = \phi_1(x)$, $x'' = \phi_2(x')$, etc.

By the properties of ϕ_1 , $x' = y'$ (in which case $x'' = y''$) or $(x', y') \in A' \cap P' \times T'$

By the properties of ϕ_2 , $x'' = y''$ or $(x'', y'') \in A'' \cap P'' \times T''$

Thus $(x, y) \in A \cap P \times T \Rightarrow \phi(x) = \phi(y) \vee (\phi(x), \phi(y)) \in A'' \cap P'' \times T''$

A similar result follows for $(x, y) \in A \cap T \times P$. \diamond

In order to define the behaviour-respecting properties of a system morphism, it is desirable to consider complete steps, since the firing of multiple transitions in the refinement may correspond to the firing of one transition in the abstraction.

Definition 3.10: Given a morphism $\phi: N \rightarrow N'$, a step Y of N is **complete** if $\forall t' \in T': \forall t \in \text{bd}(N_t): \phi(\{t\} \times Y(t)) = \{t'\} \times \phi(Y)(t')$.

Thus, a step is complete if all the border transitions occur with matching modes (which also match the mode occurrence of the corresponding abstract transition).

Definition 3.11: A **system morphism** $\phi: N \rightarrow N'$ is a mapping from N to N' which is **behaviour-respecting**, namely:

- (a) ϕ is surjective with respect to P' , T' , A'
- (b) ϕ is linear and total over both \mathbb{M} and \mathbb{Y} .
- (c) $\forall M \in \mathbb{M}_R: \forall Y \in \mathbb{Y}$:
 Y is complete and realisable as $Y_1 Y_2 \dots Y_n$ at marking $M \Rightarrow$
 $\phi(Y)$ is realisable as $\phi(Y_1)\phi(Y_2)\dots\phi(Y_n)$ at marking $\phi(M)$
- (d) $\forall M \in \mathbb{M}_R: \forall Y \in \mathbb{Y}$: Y is complete \Rightarrow
 $\phi(M + E^+(Y) - E^-(Y)) = \phi(M) + \phi(E^+)(\phi(Y)) - \phi(E^-)(\phi(Y))$

Note:

- (c) If the refined step is complete, then its realisation can be used to derive the realisation of the corresponding abstract step, by projecting or restricting each component step.
- (d) This modified rule for the effect of a refined step (cf. [18]) is used since we cannot consider the component steps (of its realisation) in isolation. Thus, part (c) guarantees the enabling of the abstract sequence, while part (d) captures its overall effect

The above definition clarifies what we mean by behaviour-respecting, also called *behavioural compatibility*. This kind of morphism is called a system morphism to emphasise that it is concerned with the behaviour of the net (following the common Petri Net distinction of a net concerning the structure, and a system including behaviour). This definition is akin to that of Winskel [18], where a Petri Net is a two-sorted algebra (the sorts being the markings and the steps) and the morphism is a homomorphism which respects the input and output operations (our E^- and E^+).

This definition captures the requirement that we identified earlier, namely that refinement constrains the behaviour of the system since refined behaviours have a corresponding abstract behaviour.

Proposition 3.12: Given two system morphisms $\phi_1: N \rightarrow N'$ and $\phi_2: N' \rightarrow N''$ their composition $\phi = \phi_2 \circ \phi_1: N \rightarrow N''$ is a system morphism.

Proof:

The composition of surjective morphisms is surjective.

The composition of total, linear mappings is total and linear.

Now, consider an arbitrary marking $M \in \mathbb{M}$, $\phi_1(M) = M' \in \mathbb{M}'$, $\phi_2(M') = M'' \in \mathbb{M}''$.

Consider an arbitrary step $Y \in \mathbb{Y}$, $\phi_1(Y) = Y' \in \mathbb{Y}'$, $\phi_2(Y') = Y'' \in \mathbb{Y}''$.

Suppose Y and Y' are complete with respect to ϕ_1 and ϕ_2 respectively.

It follows that Y is complete with respect to ϕ .

Then if Y is realisable as $Y_1 Y_2 \dots Y_n$, then Y' is realisable as $\phi_1(Y_1) \phi_1(Y_2) \dots \phi_1(Y_n)$.

Hence Y'' is realisable as $\phi_2(\phi_1(Y_1)) \phi_2(\phi_1(Y_2)) \dots \phi_2(\phi_1(Y_n))$.

Also: $\phi_2(\phi_1(M + E^+(Y) - E^-(Y)))$

$$= \phi_2(\phi_1(M) + \phi_1(E^+)(\phi_1(Y)) - \phi_1(E^-)(\phi_1(Y))) \quad \text{by properties of } \phi_1$$

$$= \phi(M) + \phi(E^+)(\phi(Y)) - \phi(E^-)(\phi(Y)) \quad \text{by properties of } \phi_2 \quad \diamond$$

The above proposition tells us that we can combine the different forms of refinement (considered in the subsequent section) and we still have a system morphism.

3.3 CPN Refinements

We now consider each of the proposed kinds of refinement. For each one, we give a formal definition, show that it satisfies our generic constraint, and then indicate how it relates to the underlying Place Transition Net (PTN). In each case, the terminology reflects the way the incremental change is used, i.e. from abstract to refined, but the morphism is always expressed as a mapping from refined to abstract.

3.3.1 Type refinement

The first, and perhaps simplest, form of refinement is to retain the structure of the net without modification but to replace some (or all) of the token and mode types by subtypes. Given our formulation of CPNs, where the arc inscriptions are functions (from modes to token multisets), it may not even be necessary to change the arc inscriptions, provided that they are given by polymorphic functions. For example, if an arc inscription is the identity function, i.e. the mode determines the token to be removed or added, then a simple change to both mode and token type would give a refined behaviour. If the arc inscription functions are *not* given by polymorphic functions, then they will need to be replaced, but the new versions must be consistent with the old. The distinctive thing about type refinement is that there is a projection function from subtype to supertype so that every refined state or action has a corresponding abstract state or action.

Definition 3.13: A morphism $\phi: N \rightarrow N'$ captures a type refinement if:

- (a) ϕ is an identity function on P, T, A , i.e. $\forall x \in P: \phi(p) = p$, etc.
- (b) $\forall x \in P \cup T: C(x) \leq: \phi(C)(x)$
- (c) $\forall x \in P \cup T: \forall c \in C(x): \phi(1^-(x, c)) = 1^-(x, \prod_{\phi(C)(x)}(c))$
- (d) $\forall (p, t) \in A: \forall (t, m) \in \mathbb{Y}: \phi(E^-(1^-(t, m)))(p) = \prod_{\phi(C)(p)}(E(p, t)(m)) = \phi(E)(p, t)(\prod_{\phi(C)(t)}(m))$
 $\forall (t, p) \in A: \forall (t, m) \in \mathbb{Y}: \phi(E^+(1^-(t, m)))(p) = \prod_{\phi(C)(p)}(E(t, p)(m)) = \phi(E)(t, p)(\prod_{\phi(C)(t)}(m))$

Note:

- (a) There is no change to the structure of the net — to places, transitions and arcs.
- (b) The colours associated with the places and transitions are consistently subtyped.
- (c) Given the consistent structure, the morphism is defined for all markings and

steps using the appropriate projection functions (see §3.1).

- (d) The arc inscriptions are consistent, i.e. the projected effect of a mode or step is the same as the effect of the projected mode or step.

Proposition 3.14: A morphism $\phi: N \rightarrow N'$ which captures a type refinement is a system morphism (in the sense of def 3.11).

Proof:

ϕ is surjective on P', T', A' , and is linear and total on \mathbb{M}, \mathbb{Y} .

Given $M \in \mathbb{M}_R$ and an enabled step $Y \in \mathbb{Y}$, i.e. $M[Y >$

Y is necessarily complete since N_t is always a single transition.

From def 3.13(c), $M' = \phi(M) \in \mathbb{M}'$ and $Y' = \phi(Y) \in \mathbb{Y}'$ are both defined

The enabling in N means $M \geq E^-(Y)$ and hence $\phi(M) \geq \phi(E^-(Y))$

Def 3.13(c, d) implies that $\phi(M) \geq \phi(E^-) \phi(Y)$ and hence Y' is enabled, i.e. $M'[Y' >$

Hence, $\phi(M - E^-(Y) + E^+(Y)) = M' - \phi(E^-)(Y') + \phi(E^+)(Y')$

Hence, $\phi(M + E^+(Y) - E^-(Y)) = M' + \phi(E^+)(Y') - \phi(E^-)(Y')$ \diamond

This result is actually stronger than that required by def 3.11 in that every enabled refined step is complete and has a corresponding enabled abstract step. In other words, given the realisation of a refined step, we can determine a realisation of the corresponding abstract step.

In §2 we gave an example of type refinement, where the book token type was extended with information about the loan type. There were corresponding changes to the relevant transition firing modes. This form of refinement can eliminate some abstract behaviour since the refined token requirements of a transition may not be satisfied, even though the abstract requirements are. Thus, the firing of the *Loan expires* transition may now be constrained by the loan type.

The underlying PTN for a CPN has one (colourless) place for each (place, colour) combination of the CPN, and one (colourless) transition for each (transition, colour) combination. If we relate this morphism to the underlying PTN (in the sense of refining the behaviour), then each value will potentially be replaced by a number of values (given the additional data components), and hence it is apparent that it corresponds to unfolding a place (transition) into a number of places (transitions). This form of net transformation (and its converse) have been classified as *unfolding* and *folding* [1].

3.3.2 Subnet refinement or extension

The second form of refinement is to add net components — places, transitions and arcs, or even additional token or mode values. As a morphism (from refined to abstract nets), this would be called a restriction, since net components are being discarded or ignored. Where token or mode types are extended, then in contrast to §3.3.1, abstraction does *not* project the additional refined values onto abstract values but rather ignores them. (In the equivalent unfolded PTN, this is the same as ignoring places, transitions and arcs.) This does not satisfy the structure-respecting requirements for a net morphism (def 3.8), but it does qualify as a system morphism (def 3.11).

Definition 3.15: A morphism $\phi: N \rightarrow N'$ captures a subnet refinement if:

- (a) The net structure is restricted, i.e. $\forall p \in P: \phi(p)$ is defined $\Rightarrow \phi(p) \in P$, and similarly for T, A .
- (b) $\forall x \in \phi(P) \cup \phi(T): C(x) \supseteq \phi(C)(x)$

- (c) $\forall x \in P \cup T: \forall c \in C(x): \phi(I^-(x,c)) = I^-(x,c)$ if $x \in \phi(P) \cup \phi(T)$, otherwise \emptyset
- (d) $\forall Y \in \mathbb{Y}: \phi(E^+(Y)) = \phi(E^+)(\phi(Y))$ and $\phi(E^-(Y)) = \phi(E^-)(\phi(Y))$

Note:

- (a) The sets of places, transitions and arcs may be restricted by ϕ .
- (b) The colours associated with retained places and transitions may be restricted.
- (c) Given the consistent colouring, the morphism is simply defined for all markings and steps.
- (d) The restricted incremental effect of the step is the same as the incremental effect of the restricted step. This implies that ignored components can refer to, but cannot permanently modify, retained components.

Proposition 3.16: A morphism $\phi: N \rightarrow N'$ which captures a subnet refinement is a system morphism (in the sense of def 3.11).

Proof:

ϕ is surjective on P', T', A' , and is linear and total on \mathbb{M}, \mathbb{Y} .

Given $M \in \mathbb{M}_R$ and an enabled step $Y \in \mathbb{Y}$, i.e. $M[Y >$

From def 3.15 (c), $M' = \phi(M) \in \mathbb{M}$ and $Y' = \phi(Y) \in \mathbb{Y}'$ are both defined

Y is necessarily complete since N_t is always a single transition.

The enabling in N means $M \geq E^-(Y)$ and hence $\phi(M) \geq \phi(E^-(Y))$

Def 3.15 (c, d) implies that $\phi(M) \geq \phi(E^-)(\phi(Y))$ and hence Y' is enabled, i.e. $M'[Y' >$

Hence, $\phi(M - E^-(Y) + E^+(Y)) = M' - \phi(E^-)(Y') + \phi(E^+)(Y')$

Hence, $\phi(M + E^+(Y) - E^-(Y)) = M' + \phi(E^+)(Y') - \phi(E^-)(Y')$ \diamond

As with type refinement, this result is stronger than that required by def 3.11 in that every enabled refined step is complete and has a corresponding enabled abstract step. In other words, given the realisation of a refined step, we can determine a realisation of the corresponding abstract step.

In §2 we gave an example of subnet refinement, where the subnet was augmented to cater for book reservations. The abstract behaviour of the net was restricted by the refinement — if a book can be borrowed given the possibility of reservations (the refined case), then it can also be borrowed if reservations are ignored (i.e. the abstract case).

If we relate this morphism to the underlying PTN, then it is immediately apparent that it corresponds to a net transformation (and its converse) which have been classified as *embedding* (or extension) and *restriction* [1].

3.3.3 Node refinement

The third form of refinement is the replacement of a place (transition) by a place (transition) bordered subnet, also called a *superplace* (*supertransition*). We refer to this as *node refinement* to distinguish it from the other forms of refinement being considered, even though traditional Petri Net theory simply refers to it as refinement [1]. The desirable properties of node refinement for CPNs have been considered elsewhere [12]. For behavioural consistency, it was argued that the subnet which refines a place should have the notion of an abstract marking, and the subnet which refines a transition should have the notion of abstract firing. These constraints were captured by requiring the morphism to be structure-respecting, colour-respecting, marking-respecting and step-respecting. Here, we propose canonical place and transition refinements, and in §4 compare this approach to the previous one. In any

case, the morphism for a node refinement is both a net morphism and a system morphism.

Definition 3.17: Given a morphism $\phi: N \rightarrow N'$ then $N_{p'}$ is a **canonical place refinement** of $p' \in P'$ provided:

- (a) $\forall p \in P - P_{p'}: \forall t \in T - T_{p'}: \phi(p) = p \wedge \phi(t) = t \wedge$
 $(p, t) \in A \Rightarrow ((p, t) \in A' \wedge E(p, t) = E'(p, t)) \wedge$
 $(t, p) \in A \Rightarrow ((t, p) \in A' \wedge E(t, p) = E'(t, p))$
 - (b) $\forall t \in \text{env}(N_{p'}): E'(p', t) = \sum_{(p, t) \in A \cap (P_{p'} \times T)} E(p, t) \text{ and } E'(t, p') = \sum_{(t, p) \in A \cap (T \times P_{p'})} E(t, p)$
 - (c) $\text{bd}(N_{p'}) = \{\text{inp1}, \text{inp2}, \dots, \text{out1}, \text{out2}, \dots\}$
 - (d) $P_{p'} = \text{bd}(N_{p'}) \cup \{\text{buf}\} \cup P_{\text{other}}$
 - (e) $T_{p'} = \{\text{accept1}, \text{accept2}, \dots, \text{offer1}, \text{offer2}, \dots\} \cup T_{\text{other}}$
 - (f) $A_{p'} = \{(\text{inp1}, \text{accept1}), (\text{accept1}, \text{buf}), \dots, (\text{buf}, \text{offer1}), (\text{offer1}, \text{out1}), \dots\} \cup A_{\text{other}}$
 - (g) $\bullet \text{inp1} \subseteq \text{env}(N_{p'}) \wedge \text{inp1} \bullet = \{\text{accept1}\} \wedge \dots$
 $\text{out1} \bullet \subseteq \text{env}(N_{p'}) \wedge \bullet \text{out1} = \{\text{offer1}\} \wedge \dots$
 $\bullet \text{buf} = \{\text{accept1}, \dots\} \wedge \text{buf} \bullet = \{\text{offer1}, \dots\}$
 - (h) $\forall x \in P_{p'} - P_{\text{other}} \cup T_{p'} - T_{\text{other}}: C_{p'}(x) = C(x) = C'(p')$
 - (i) $\forall a \in A_{p'} - A_{\text{other}}: E_{p'}(a) = \text{Id}$
 - (j) $\forall p \in P: \forall c \in C(p): \phi(1^-(p, c)) = 1^-(p', c) \text{ if } p \in P_{p'} - P_{\text{other}}$
 $= \emptyset \text{ if } p \in P_{\text{other}}$
 $= 1^-(p, c) \text{ otherwise}$
- $\forall t \in T: \forall c \in C(t): \phi(1^-(t, c)) = 1^-(t, c) \text{ if } t \notin T_{p'} \text{ otherwise } \emptyset$

Note:

- (a) Apart from the subnet $N_{p'}$ and its incident arcs, the rest of the net is unchanged.
- (b) The flow of tokens across the boundary of the place refinement is consistent.
- (c) The border of the subnet consists of the places $\text{inp1}, \dots, \text{out1}, \dots$
- (d-f) The component places, transitions and arcs consist of the basis places, transitions and arcs respectively, plus others which constitute the subnet refinement of the *accept* and *offer* transitions (cf. fig 2.4).
- (g) The input (output) border places only have input (output) from (to) environment transitions, other arcs incident on basis places are exactly those of part (e).
- (h) The colour of the basis places and transitions are all the same.
- (i) The arc inscription for all the basis arcs is the identity function.
- (j) In abstracting the place refinement, the abstract marking is given by the marking of the basis places and the internal actions are ignored.

Proposition 3.18: A morphism $\phi: N \rightarrow N'$ with a canonical place refinement constitutes a system morphism.

Proof:

ϕ is surjective on P', T', A' , and is linear and total on \mathbb{M}, \mathbb{Y} .

Given $M \in \mathbb{M}_R$ and an enabled step $Y \in \mathbb{Y}$, i.e. $M[Y >$

From def 3.17 (j), $M' = \phi(M) \in \mathbb{M}'$ and $Y' = \phi(Y) \in \mathbb{Y}'$ are both defined

Y is necessarily complete since N_t is always a single transition.

Given the linearity of ϕ , $M \geq E^-(Y)$ implies $\phi(M) \geq \phi(E^-(Y))$

From def 3.17 (a, b), $\phi(M) \geq \phi(E^-)(\phi(Y))$ and hence Y' is enabled, i.e. $M'[Y' >$

Hence, $\phi(M - E^-(Y) + E^+(Y)) = \phi(M) - \phi(E^-)(Y') + \phi(E^+)(Y')$

Hence, $\phi(M + E^+(Y) - E^-(Y)) = \phi(M) + \phi(E^+)(Y') - \phi(E^-)(Y')$

◇

As with type and subnet refinement, this result is stronger than that required by def 3.11 in that every enabled refined step is complete and has a corresponding enabled abstract step. In other words, given the realisation of a refined step, we can determine a realisation of the corresponding abstract step. This is not the case for the following transition refinement.

Definition 3.19: Given a morphism $\phi: N \rightarrow N'$ then N_t is a **canonical transition refinement** of $t' \in T'$ provided:

- (a) $\forall p \in P - P_t: \forall t \in T - T_t: \phi(p) = p \wedge \phi(t) = t \wedge$
 $(p, t) \in A \Rightarrow ((p, t) \in A' \wedge E(p, t) = E'(p, t)) \wedge$
 $(t, p) \in A \Rightarrow ((t, p) \in A' \wedge E(t, p) = E'(t, p))$
- (b) $\forall p \in \text{env}(N_t): E'(p, t') = \sum_{(p, t) \in A \cap (P \times T_t)} E(p, t) \text{ and } E'(t', p) = \sum_{(t, p) \in A \cap (T_t \times P)} E(t, p)$
- (c) $\text{bd}(N_t) = \{\text{inp1}, \dots, \text{out1}, \dots\}$
- (d) $P_t = \{\text{recd1}, \dots, \text{send1}, \dots\} \cup P_{\text{other}}$
- (e) $T_t = \text{bd}(N_t) \cup \{\text{switch}\} \cup T_{\text{other}}$
- (f) $A_t = \{(\text{inp1}, \text{recd1}), (\text{recd1}, \text{switch}), \dots, (\text{switch}, \text{send1}), (\text{send1}, \text{out1}), \dots\} \cup A_{\text{other}}$
- (g) $\bullet \text{inp1} \subseteq \text{env}(N_t) \cup P_{\text{other}} \wedge \bullet \text{recd1} = \{\text{inp1}\} \wedge \text{recd1} \bullet = \{\text{switch}\} \wedge \dots$
 $\text{out1} \bullet \subseteq \text{env}(N_t) \cup P_{\text{other}} \wedge \text{send1} \bullet = \{\text{out1}\} \wedge \bullet \text{send1} = \{\text{switch}\} \wedge \dots$
 $\bullet \text{switch} = \{\text{recd1}, \dots\} \wedge \text{switch} \bullet = \{\text{send1}, \dots\}$
- (h) $\forall x \in P_t - P_{\text{other}} \cup T_t - T_{\text{other}}: C_t(x) = C(x) = C'(t')$
- (i) $\forall a \in A_t - A_{\text{other}}: E_t(a) = \text{Id}$
- (j) $\forall p \in P_t - P_{\text{other}}: M_{0t}(p) = \emptyset$
- (k) $\forall p \in P: \forall c \in C(p): \phi(1'(p, c)) = 1'(p, c) \text{ if } p \in P - P_t, \text{ otherwise } \emptyset$
 $\forall t \in T: \forall c \in C(t): \phi(1'(t, c)) = 1'(t', c) \text{ if } t = \text{switch}$
 $= \emptyset \text{ if } t \in T_t - \{\text{switch}\}$
 $= 1'(t, c) \text{ otherwise}$

Note:

- (a) Apart from the subnet N_t and its incident arcs, the rest of the net is unchanged.
- (b) The flow of tokens across the boundary of the transition refinement is consistent.
- (c) The border of the subnet consists of the transitions *inp1*, ..., *out1*, ...
- (d-f) The component places, transitions and arcs consist of the basis places, transitions and arcs respectively, plus others which constitute the subnet refinement of the border transitions (cf. fig 2.6).
- (g) The input (output) border transitions only have input (output) from (to) environment places or the other internal places, the basis places only have the incident arcs of part (e).
- (h) The colour of the basis places and transitions are all the same.
- (i) The arc inscription for all the basis arcs is the identity function.
- (j) The initial marking of all basis places is empty.
- (k) The internal marking of the supertransition is ignored by the abstraction and the firing of the *switch* transition corresponds to the abstract firing of t' .

Proposition 3.20: A morphism $\phi: N \rightarrow N'$ with a canonical transition refinement constitutes a system morphism.

Proof:

ϕ is surjective on P', T', A' , and is linear and total on \mathbb{M}, \mathbb{Y} .

Given $M \in \mathbb{M}_R$ and a complete step $Y \in \mathbb{Y}$.

From def 3.19 (k), $M' = \phi(M) \in \mathbb{M}'$ and $Y' = \phi(Y) \in \mathbb{Y}'$ are both defined

Suppose Y is realisable as $Y_1 Y_2 \dots Y_n$ at M

Consider $\phi(Y_1) \phi(Y_2) \dots \phi(Y_n)$ and, in particular, an arbitrary element $\phi(Y_i)$

Transitions external to N_i occur in $\phi(Y_i)$ exactly as in Y_i and are similarly enabled.

Transitions internal to N_i apart from *switch* occurring in Y_i are ignored in $\phi(Y_i)$

The transition *switch* in N_i occurring in Y_i occurs as t' in $\phi(Y_i)$

Completeness guarantees that border transitions occur with the same mode(s) as t'

The canonical construction guarantees that the input tokens are consumed first.

Thus, transition t' is enabled in $\phi(Y_i)$

Finally, defs 3.10, 3.19(b) guarantee that the total effect of Y and Y' are the same

Hence, $\phi(M + E^+(Y) - E^-(Y)) = \phi(M) + \phi(E^+)(Y') - \phi(E^-)(Y')$ \diamond

Definition 3.21: A morphism $\phi: N \rightarrow N'$ **captures a canonical node refinement** if ϕ has a canonical place or transition refinement.

In §2 we gave examples of node refinement, one where a place was replaced by a subnet to indicate the more detailed processing of a book on its return from loan, and one where a transition was replaced by a subnet to indicate more of the details of borrowing a book. Again, such refined behaviour had corresponding abstract behaviour, but not necessarily vice versa.

If we relate this morphism to the underlying PTN then it corresponds to the refinement of a place or transition by an appropriately-bounded subnet. This form of net transformation (and its converse) have been classified as *refinement* and *abstraction* [1].

4 Properties of Abstractions

The previous section has defined three forms of CPN refinement which maintain behavioural compatibility. In this section we show that even though they appear to be quite constrained, they are at least as general as the earlier proposals [12].

The earlier proposals required that the flow of tokens across the boundaries of node refinements should be consistent with the corresponding abstractions:

Definition 4.1: A structure-respecting morphism $\phi: N \rightarrow N'$ is **colour-respecting** if C', E' of N' satisfy:

- (a) $\forall x' \in P' \cup T': \forall x \in \text{bd}(N_x): C(x) \leq C'(x')$
- (b) $\forall (x', y') \in A': E'(x', y') = \sum_{x \in \text{bd}(N_{x'})} \sum_{y \in \text{bd}(N_{y'})} E(x, y)$

Note:

- (a) The colour of a superplace or a supertransition must be compatible with the colours of the corresponding places or transitions in its border.
- (b) The morphism preserves the token transfer across the interfaces between superplaces and supertransitions.

The above property is satisfied by the node refinements proposed in §3.3.3 as is apparent from defs 3.17 (b) and 3.19 (b). The propositions which follow will guarantee the converse, namely that the colour-respecting morphisms we consider have corresponding canonical node refinements.

In the earlier paper [12], it was argued that behavioural compatibility required that place refinements should have an associated abstract marking which was invariant over the internal actions of the subnet. This was captured by the following property.

Definition 4.2: A colour-respecting morphism $\phi: N \rightarrow N'$ is **marking-respecting** if:

- (a) ϕ is linear and total on \mathbb{M}
- (b) $\forall p' \in P': \forall M \in \mathbb{M}: \phi(M)(p') = \sum_{p \in N_{p'}} \phi(M)(p)$
- (c) $\forall p' \in P': M_1[(t, c) > M_2:$

$$\phi(M_2)(p') = \phi(M_1)(p') - \sum_{p \in \text{bd}(N_{p'})} E(p, t)(c) + \sum_{p \in \text{bd}(N_{p'})} E(t, p)(c) \text{ if } t \in \text{env}(N_{p'})$$

$$= \phi(M_1)(p') \text{ otherwise}$$

Note:

- (a) For regularity, ϕ is linear and defined over all components of \mathbb{M} , not just reachable markings, which would be the minimal requirement.
- (b) The abstract marking of a superplace is determined by the refined markings of its constituent places. This is largely implied by requirement (c) but it is clearer to state it explicitly.
- (c) The abstract marking of a superplace is only modified by environment transitions and not by other external transitions or internal transitions.

The following propositions elucidate the relationship between marking-respecting morphisms and the canonical place refinements of §3.3.3.

Proposition 4.3: A morphism $\phi: N \rightarrow N'$ with a canonical place refinement (as in def 3.17) is marking-respecting.

Proof: This follows directly from def 3.17(j). \diamond

Proposition 4.4: An arbitrary marking-respecting refinement can be transformed into an behaviourally compatible canonical place refinement where the internal place *buf* is redundant.

Proof:

Build the basis of a canonical place refinement as follows:

- for each border place with one or more input arcs from the environment, introduce a new input border place which usurps those input arcs;
- from each new input border place, add an arc to a corresponding *accept* transition (which is newly introduced), and add an arc from this *accept* transition to the original border place;
- for each border place with one or more output arcs to the environment, introduce a new output border place which usurps those output arcs;
- from each such original border place, add an arc to a corresponding *offer* transition (which is newly introduced), and add an arc from this *offer* transition to the new output border place;
- add a new place *buf* with input arcs from all the newly introduced *accept* transitions and output arcs to all the newly introduced *offer* transitions;
- the token and mode type of all introduced places and transitions is that of the token type for the abstract place, and the new arcs all have the same identity function as inscription.

This process is illustrated in fig 4.1, where the original components are shaded:

- the border place *b* (which has both input and output arcs with the environment) is replaced by two border places *ib* (for input) and *ob* (for output)
- there is an *accept* transition *ab* corresponding to the new input border place *ib*

- there is an *offer* transition ob corresponding to the new output border place ob
- the input arc from the environment incident on b is now incident on ib
- the output arc to the environment incident on b is now incident on ob
- there is a new input arc incident on b and a new output arc incident on b

Similarly for the other border places a and c .

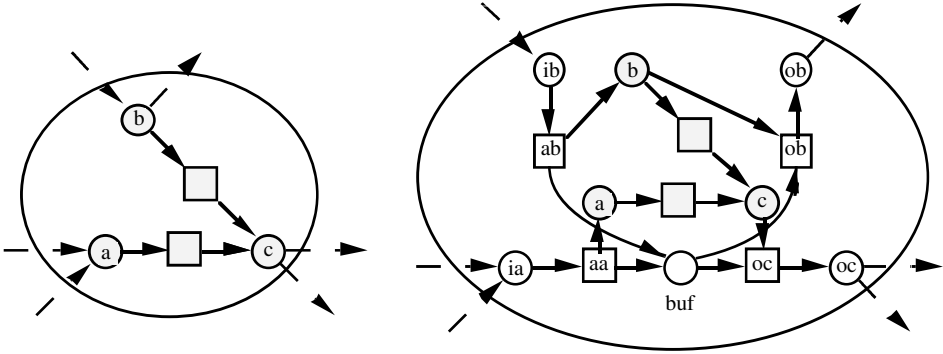


Fig 4.1: Arbitrary place refinement

The behaviour of the transformed net is compatible with the original (see def 3.11):

- as soon as a token is deposited into a new input border place, the corresponding *accept* transition can fire, depositing a copy of the token into the corresponding original border place, and a copy is also added to the place *buf*
- as soon as a token is available in one of the original border places with an output arc to the environment, then the corresponding *offer* transition can be fired, making the token available in the new border place (and thence to the environment)
- given a transition sequence of the modified net, the corresponding original sequence can be extracted by ignoring the firing of *accept* and *offer* transitions.

Note that the place *buf* is redundant (and therefore could be eliminated):

- when an environment transition of the original place refinement deposited one or more tokens into the border places, the abstract marking given by $\phi(M)(p')$ varied by exactly those tokens (see def 4.2(c)).
- with the new refinement, the same property holds (except that we now have new input border places).
- the removal of tokens from any new input border place (by firing the corresponding *accept* transition) is matched by depositing the token in the *buf* place and in the corresponding original border place.
- the firing of the original internal transitions did not affect the abstract marking, and hence it stays synchronised with the marking of place *buf* (see def 4.2(c)).
- finally, the removal of tokens from the original place refinement is now matched by the removal of tokens from the place *buf*

Finally, it is straightforward to check that the original subnet has become a subnet refinement augmenting the *accept* and *offer* transitions of the basis for a canonical place refinement. \diamond

Again, the earlier paper [12] argued that behavioural compatibility required that transition refinements should have the notion of abstract firing. This was captured by the following property.

Definition 4.5: A colour-respecting morphism $\phi: N \rightarrow N'$ is **step-respecting** if

- (a) ϕ is linear on \mathbb{Y}
- (b) if $Y \in \mathbb{Y}$ is realisable at $M \in \mathbb{M}$, and $\phi(Y)$ is defined with $\phi(Y) = Y' \in \mathbb{Y}'$ realisable at $\phi(M) = M' \in \mathbb{M}'$, then

$$\forall t \in \text{bd}(N_t): Y(t) = Y'(t')$$
- (c) if $Y \in \mathbb{Y}$ is realisable at $M_1 \in \mathbb{M}$ leading to M_2 , and $\phi(Y)$ is defined with $\phi(Y) = (t', c') \in \mathbb{Y}'$ realisable at $\phi(M_1) = M_1' \in \mathbb{M}'$, then:

$$\begin{aligned} M_2(p) &= M_1(p) - \sum_{t \in \text{bd}(N_{t'})} E(p, t)(c') + \sum_{t \in \text{bd}(N_{t'})} E(t, p)(c') & \text{if } p \in \text{env}(N_{t'}) \\ &= M_1(p) & \text{if } p \in P - P_{t'} \end{aligned}$$

Note:

- (a) ϕ is linear, i.e. $\phi(Y_1 + Y_2) = \phi(Y_1) + \phi(Y_2)$ provided $\phi(Y_1), \phi(Y_2)$ are defined.
- (b) The occurrence of border transitions is determined by the abstract firing modes. This is largely implied by requirement (c) but it is clearer to state it explicitly.
- (c) Of the places external to the supertransition, only the environment places are modified and only by the effect of the border transitions.

Proposition 4.6: Given a morphism $\phi: N \rightarrow N'$ with a canonical transition refinement N_t , a realisable step Y of N has a corresponding realisable step of N' if and only if it is complete.

Proof:

Consider a complete step Y realisable as the sequence Y^* :

In Y^* , each abstract firing of $N_{t'}$ will include a firing of the transition *switch*.

The input border transitions of $N_{t'}$ will fire prior to the firing of *switch*.

The consumed tokens could also be consumed later — just before the *switch*.

The output border transitions of $N_{t'}$ will fire after the firing of *switch*.

The generated tokens could be generated earlier — just after the *switch*.

Hence, the complete firing of $N_{t'}$ could be replaced by the abstract firing of t'

Specifically, the firing of t' could replace the *switch* transition in the sequence.

Conversely, suppose a refined sequence is not complete:

Then the firing of some border transition is not included.

The corresponding token transfers will not occur.

Hence, the effect on the environment will not match that of t' . \diamond

The following propositions elucidate the relationship between step-respecting morphisms and the canonical transition refinements of §3.3.3.

Proposition 4.7: A morphism $\phi: N \rightarrow N'$ with a canonical transition refinement is step-respecting.

Proof:

Every refined step has a corresponding abstract step.

If the refined step is complete and realisable, then so is the abstract step (prop 4.6).

The construction ensures that all border transition have the same occurrence modes

Hence property (b) of def 4.5 holds.

Property (c) then follows from def 3.19 (a,b). \diamond

Proposition 4.8: An arbitrary step-respecting transition refinement with distinct input and output border transitions can be transformed into a behaviourally compatible canonical transition refinement.

Proof:

An arc to a corresponding *recd* place is added from each input border transition.

An arc from a corresponding *send* place is added to each output border transition.

The subnet is augmented by a *switch* transition.

The subnet is augmented by arcs from the *recd* places to the *switch* transition.

The subnet is augmented by arcs from the *switch* transition to the *send* places.

Behavioural compatibility can be demonstrated by ignoring the firing of *switch*.

It is straightforward to check that the original subnet is a subnet refinement augmenting the *accept* and *offer* transitions of the basis for a canonical place transition. \diamond

The above results demonstrate that the current proposals, while being more prescriptive, are just as general as the earlier proposals.

5 Conclusions and Further Work

This paper has significantly extended a previous one on the abstraction of CPNs [12]. The context is that of refining CPNs while maintaining behavioural compatibility. We have identified three kinds of refinement, namely *type*, *subnet* and *node* refinement. In practical applications, it is anticipated that these will be used in concert. Further, we have shown that by considering node refinement in conjunction with subnet refinement, it was possible to propose a canonical form for node refinement which could be extended by subnet refinement to be behaviourally compatible with an arbitrary node refinement (as in [12]).

We have shown that the proposed refinements all qualify as system morphisms, and that the composition of system morphisms is also a system morphism. This notion of morphism is an extension of that proposed by Winskel [18]. By identifying the three kinds of refinement, we have proposed an answer to the question posed by Winskel, namely the appropriate kind of morphism for CPNs.

There are a number of interesting avenues for further work. Given the generality of these canonical forms, we propose to incorporate them into a Petri Net tool, thereby supporting the designer in producing well-structured specifications. By imposing behavioural constraints on abstraction, it is possible to use the abstraction to prove properties about the modelled system, which is often important for large complex systems with excessively large state spaces. We are continuing to investigate the range of applicability of these forms of refinement to practical problems [11]. We are also developing incremental reachability algorithms to take advantage of the behavioural compatibility [14]. Further down the track, we would like to identify when refinements are in some sense independent and hence can be analysed in isolation.

Acknowledgements

The author is pleased to acknowledge the helpful discussions held with Robert Esser, Glenn Lewis and Joachim Wehler. The careful comments of the reviewers have helped to improve this paper significantly.

References

- [1] B. Baumgarten *Petri-Netze: Grundlagen und Anwendungen* Wissenschaftsverlag (1990).
- [2] L. Cardelli and J.C. Mitchell *Operations on Records* Mathematical Foundations of Programming Semantics, Lecture Notes in Computer Science 442, Springer Verlag (1989).

- [3] J. Desel and A. Merceron *Vicinity Respecting Net Morphisms* Advances in Petri Nets 1990, G. Rozenberg (ed.), Lecture Notes in Computer Science 483, pp 165-185, Springer-Verlag (1990).
- [4] J. Desel and A. Merceron *Vicinity Respecting Homomorphisms for Abstracting System Requirements* Report 337, Universität Karlsruhe (1996).
- [5] A. Diller *Z: An Introduction to Formal Methods* 2nd edn., Wiley (1994).
- [6] R. Fehling *A Concept of Hierarchical Petri Nets with Building Blocks* Advances in Petri Nets 1993, G. Rozenberg (ed.), Lecture Notes in Computer Science , pp 148-168, Springer-Verlag (1993).
- [7] H.J. Genrich *A Dictionary of Some Basic Notions of Net Theory* Net Theory and Applications, W. Brauer (ed.), Lecture Notes in Computer Science 84, pp 519-535, Springer-Verlag (1980).
- [8] H.J. Genrich *Predicate/Transition Nets* Advances in Petri Nets 1986 – Part 1, W. Brauer, W. Reisig, and G. Rozenberg (eds.), Lecture Notes in Computer Science 254, Springer-Verlag (1987).
- [9] K. Jensen *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use – Volume 1: Basic Concepts* EATCS Monographs in Computer Science, Vol. 26, Springer-Verlag (1992).
- [10] K. Jensen, S. Christensen, P. Huber, and M. Holla *Design/CPN™: A Reference Manual* MetaSoftware Corporation (1992).
- [11] C. Lakos and G. Lewis *A Catalogue of Incremental Changes for Coloured Petri Nets* Technical Report TR99-02, Department of Computer Science, University of Adelaide (1999).
- [12] C.A. Lakos *On the Abstraction of Coloured Petri Nets* Proceedings of 18th International Conference on the Application and Theory of Petri Nets, Lecture Notes in Computer Science 1248, pp 42-61, Toulouse, France, Springer-Verlag (1997).
- [13] C.A. Lakos *The Compositionality of Abstraction for Coloured Petri Nets* Technical Report TR99-01, Department of Computer Science, University of Adelaide (1999).
- [14] G. Lewis and C. Lakos *Incremental Reachability Algorithms* Technical Report TR99-1, Department of Electrical Engineering and Computer Science, University of Tasmania (1999).
- [15] J. Meseguer and U. Montanari *Petri Nets are Monoids* Information and Computation, **88**, 2, pp 105-155 (1990).
- [16] C.A. Petri *Introduction to General Net Theory* Net Theory and Applications, W. Brauer (ed.), Lecture Notes in Computer Science 84, pp 1-19, Springer-Verlag (1980).
- [17] W. Reisig *Petri Nets in Software Engineering* Advances in Petri Nets 1986 – Part 2, W. Brauer, W. Reisig, and G. Rozenberg (eds.), Lecture Notes in Computer Science 255, pp 63–96, Springer-Verlag (1987).
- [18] G. Winskel *Petri Nets, Algebras, Morphisms, and Compositionality* Information and Computation, **72**, pp 197-238 (1987).

Modelling and Analysis of a DANFOSS Flowmeter System Using Coloured Petri Nets

n n n l n n
u u
u
{louisel,kris}@daimi.au.dk

Abstract. DANFOSS u u
DANFOSS fl u
u fl k
u u fl
u u
u u
u k
u fl
fl

1 Introduction

n DANFOSS n l n l n n
k n n l l n n n
n n n l n l
n n ll n n l DANFOSS fl
n DANFOSS INSTRUMENTATION
DANFOSS n
l l k n n fl
l n fl n l
n n n fl n
n n l l n
n DANFOSS n n fl
n l l n fl l n l n
n n ll fl
n
l k n l n l n

fl n n l n n l n n
 n n n n DANFOSS n
 l k n l n n
 n n
 ll n n n
 l n n n
 n l n n l n
 l k n fl n
 l n n n l n l
 n l ll n l n k
 l n n n l n n l n nk
 n n n n l l l l
 n l n fl
 DANFOSS
 n ll n n
 n n n n n n
 DANFOSS fl n n l
 l fl n
 n n fl n n
 n l n n
 n n ll n l n
 l l l n
 l ll n l

2 Overview of the Project

n n n n l n n DANFOSS
 n l n l n n
 n n l n n
 fl ll n n
 l l n n n
 n n l n n
 n l n n
 n n n l
 fl n n l l n n
 l fl n n n
 n n DANFOSS n l n n l n n
 fl l l DANFOSS
 n n n fl n
 ll n l n n fl n

u
 n n n n n l
 n l n fl n
 n n l l n
 n l n l n
 l l ll n
 n n l n n
 l ll n n ll
 n l k l n n n
 n l n n n
 l n l n
 n n n n n
 n l n n ll l
 l n n l n
 l l n l n
 l n DANFOSS ll
 l l n l
 n n DANFOSS
 n n ll n
 n l n n n
 n l l n DANFOSS
 n n l n
 n n *state explosion problem*
 l n n l n n fl
 n n n l ll n n
 fl n l k n l n n
 n n l n fl
 n n l l n
 n n l ll n n fl
 l l n n fl l
 l n n n
 ll l n l n l n n
 n l l n
 l n l fl n
 l fl

3 The DANFOSS Flowmeter System

n n DANFOSS fl ll n
 ll n

fl n n l n n n
 fl n n ll l l l
 fl l l l

3.1 System Architecture and Communication Protocols

ll fl fl
 n n modules nn Controller Area Net-
 work Applications l n n ll CAN
 Applications n driver n l
 n l fl n n l
 n n n n l n n ll
 attributes ll l l l n n ll
 n n n n n n
 n n ll
 n

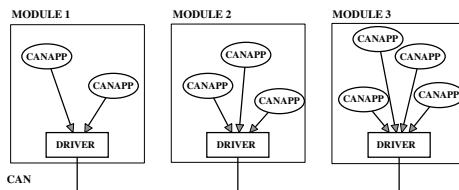


Fig. 1.

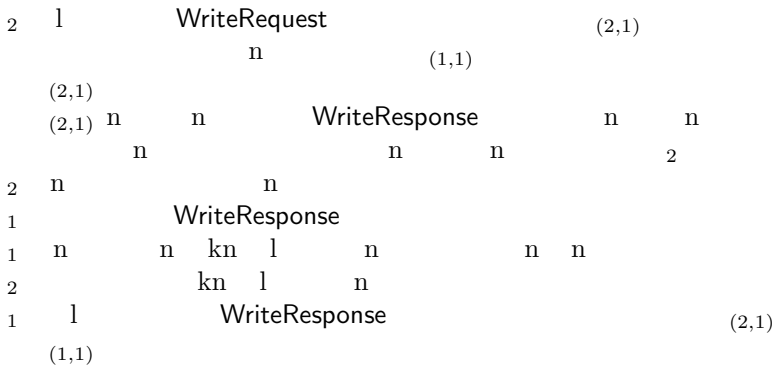
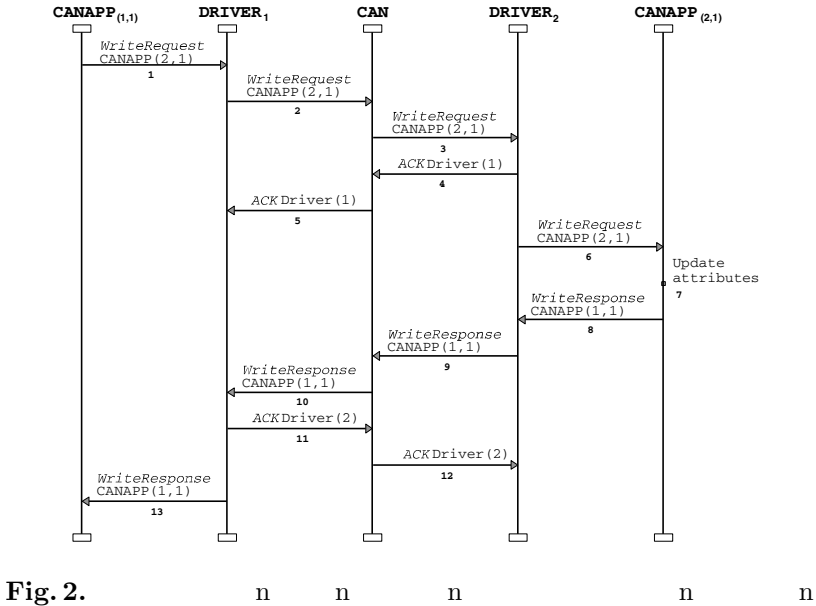
fl l n l n n n l
 n n n l l l
 l fl n
 n n n l n
 l n
 n l n l n
 n l j n l i l l ll n n (i,j)
 n l j n l j n l j n
 n n n n l
 l l n n ll l nk n l n l
 ll n n n fl n n n
 ll ll n n n l n l
 n n n l n n ll
 ll n n n

u

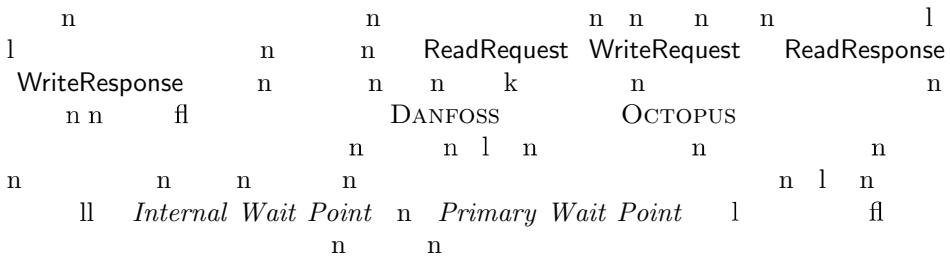
Table 1. n n n

	u
u	u u
	u u u
u	u u
	u u
	u u k
	u u k

Read ReadRequest n ReadResponse n *Write*
WriteRequest n WriteResponse n n n n n
n n n n n n n n n
ReadRequest WriteRequest n n n
ReadResponse WriteResponse n n n
n n n l n
l n n n n l
n n n n n n kn l n
n n n n n Broadcast n Event
n n n n ll n fl
Broadcast n l
n Event n l
l n n n n fl n n
l n n l n n n
ll n n n l n l n
n l n l kn n n l n
n
n n ll n
l n n ll n n
n n fl n n n n
n n ll n n n (2,1)
n n l (1,1) n ll n n n
n n k n l n
n n WriteRequest n (2,1)
(1,1) l n n 1
1 n n
2
2 n n kn l n n n
1 kn l n



3.2 CANAPP Design Patterns



Internal Wait Point (IWP) approach.



u

l k l n ll n l
n l n l n n
n
Primary Wait Point (PWP) approach. n n
n n n ll
n n l n l k n
n n n n
l n l n
ll l n n
n n l DANFOSS n ll
n l k l l n l
n l n fl
n n n n l ll n
n n l n n n l
Absence of Deadlocks. l n l n fl
n n n ll n fl
l k
Absence of Attribute Corruption. n n
fl n n l
n
Topology Independence. n n n n n fl
l l n
n fl l n n n l
n l n
l n n n l

4 CPN Model of the Flowmeter System

n n l l fl
n l n n l l n
n n l l n l n
n n n n l n
n n n n n l n
n n n n n n l
kn n n l n n n n
fl n n l n nl n l n
n l kn

4.1 CPN Model Overview

ll n l n n l

n n l pages n n n
 l n n n n n
 n n ll substitution transition n
 n n n n

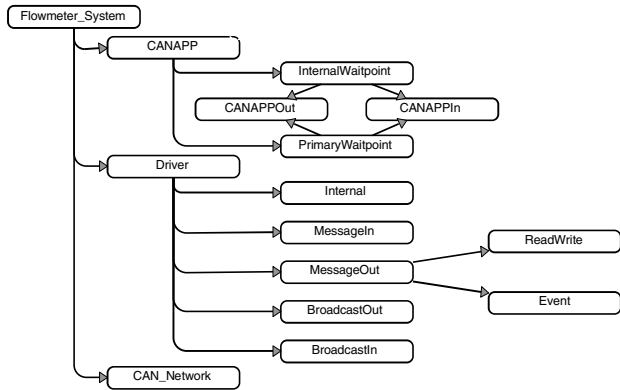


Fig. 3.

l n n n ll n
 n n CANAPP PrimaryWaitPoint InternalWaitPoint CANAP-
 PIn n CANAPPOut n ll n n n
 Driver Internal MessageIn MessageOut ReadWrite Event BroadcastOut
 n BroadcastIn ll n n CAN_Net-
 work
 Flowmeter_System n
 l n n n n n l n
 n n n n n l
 l l fl n l
 n l ll n n l l
 CAN_Network Driver n CANAPP ll n l n n
 n n n n ll n ll l n n
 l l ll n n n ll n
 ll n l n l

4.2 Modelling of the CANAPPs

 PrimaryWaitPoint n
 l n n n
 ll n n l n n ll
 n l n n n ll n l n
 l n n n CANAPPOut n n ll n

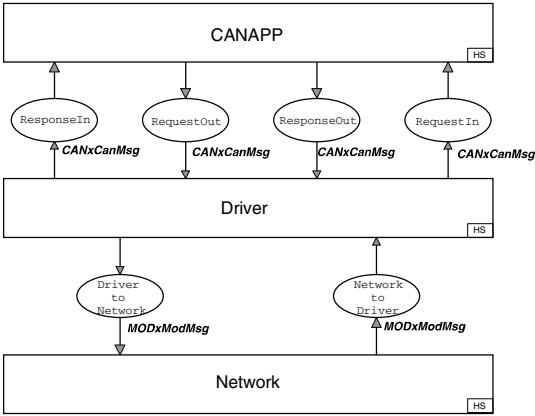


Fig. 4. Flowmeter_System

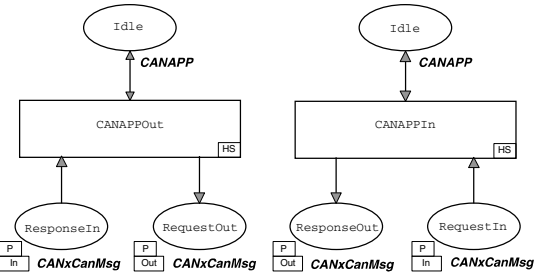
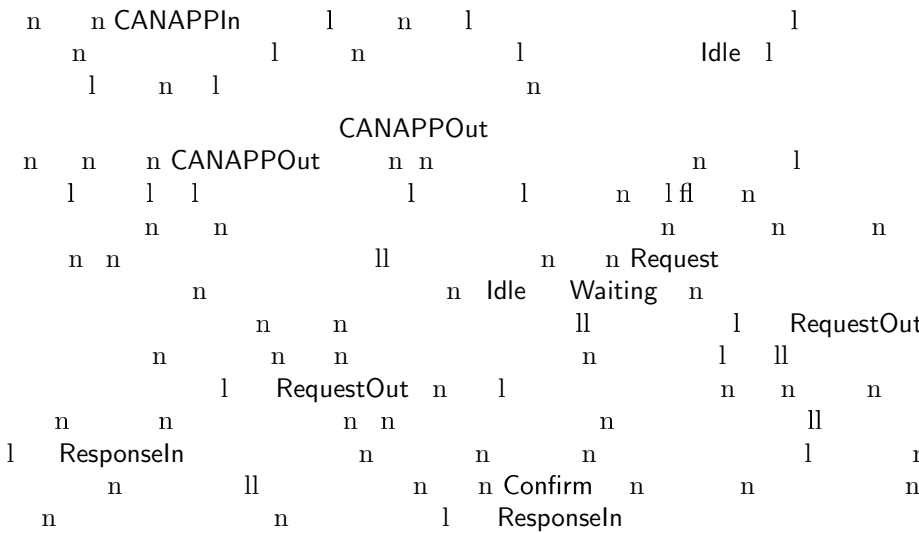


Fig. 5. PrimaryWaitPoint



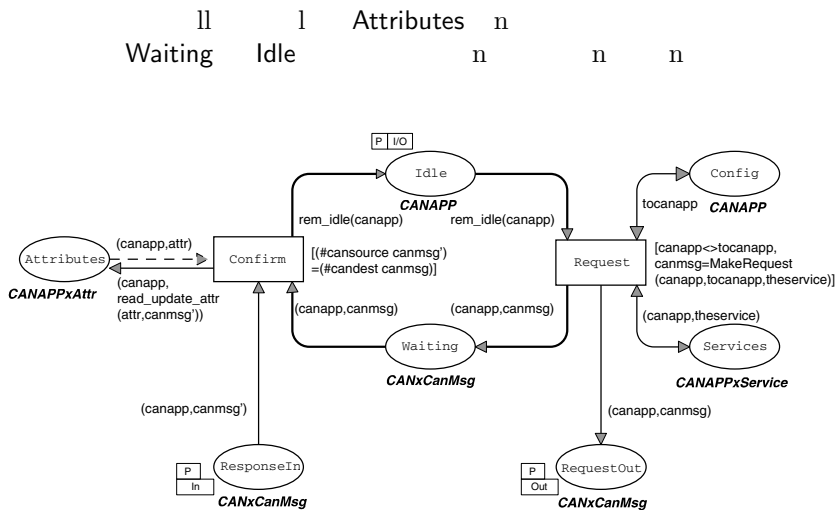
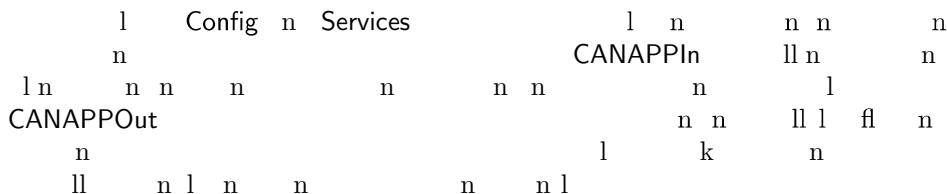
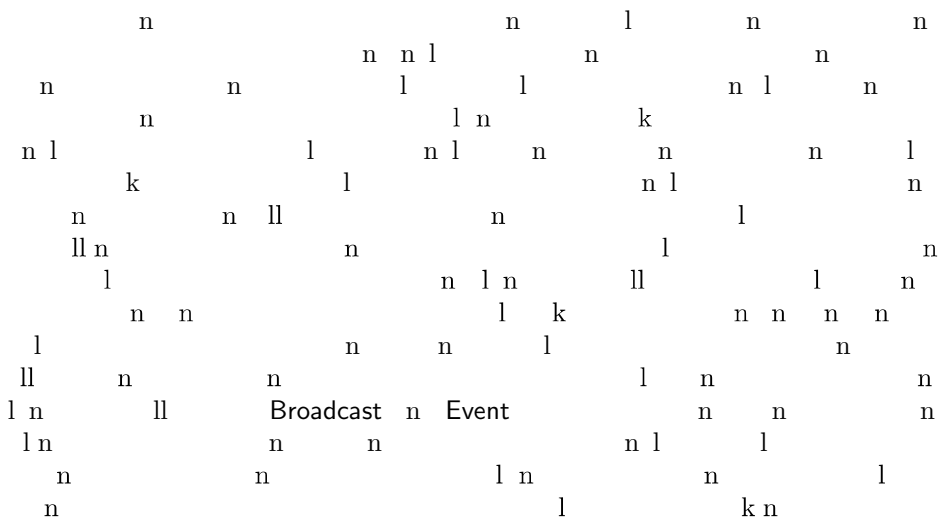


Fig. 6. CANAPPOut



5 Analysis of Two Initial Design Proposals



5.1 Analysis Goals

$$\begin{array}{ccccccc}
 & & l & & n & l & n \\
 n & & l & l & ll & & n \\
 & n & & n & & n & n \\
 & & l & & k & l & l \\
 queries & n & & l & n & & n \\
 & n & & l & n & & n \\
 & & n & l & & & n \\
 n & k & n & & l & & n
 \end{array}$$

Absence of Deadlocks.

$$\begin{array}{ccccccc}
 & & l & dead & markings & n & l \\
 n & l & n & n & n & n & l \\
 & l & & l & & l & n \\
 Markings & l & & l & & k & n
 \end{array}$$

Absence of Attribute Corruption.

$$\begin{array}{ccccccc}
 n & ll & & n & l & n & n \\
 n & l & & n & & n & n \\
 & l & & n & & n & l \\
 n & n & n & n & l & l & n \\
 ASK-CTL & & k & & l & k & l \\
 n & n & & n & & & n \\
 & n & & (i,j) & n & ll & n \\
 l & n & l & n & n & n & l
 \end{array}$$

$$\begin{array}{l}
 AG \text{ Request}, \langle canapp \quad (i,j) \rangle \Rightarrow \\
 A \neg \text{Indication}, \langle canapp \quad (i,j) \rangle \quad U \text{ Confirm}, \langle canapp \quad (i,j) \rangle
 \end{array}$$

$$\begin{array}{ccccccc}
 & l & & n & n & AG & n \\
 n & n & n & n & (i,j) & n & n \\
 n & \text{Indication} & nn & n & n & n & n \\
 n & U & n & n & \text{Confirm} & n & n \\
 & (i,j) & n & n & n & n & n \\
 n & & (i,j) & n & n & \text{Request}, \langle canapp & (i,j) \rangle \\
 & n & n & \text{Indication} & n & \text{Confirm} & n \\
 & n & n & \text{Request} & l & & n \\
 n & n & \text{Indication} & l & n & l & n \\
 n & n & n & \text{Confirm} & l & n & n
 \end{array}$$

Topology Independence.









$$\begin{array}{ccccccc}
 n & n & n & l & & l & l \\
 & & n & & n & n & l \\
 n & fl & & n & n & n & n
 \end{array}$$

n l n n n l k n l n l
n n n n n n n n n
n l k n

5.2 Analysis Results

n n n n n n n n n
l n n n n n n n n
n n n Configuration l n n n n n n
n l n n n n n n n
n n n n n n l
n n n l n n n
l WP l n n n n
Nodes n Arcs l n n n n n n
l Time l n n n k n
ll n n n k n
n ll
n n n
l k n n n n
l n n n l n l
n l n ll l n n n
n n n nl n l n n
n n n l n n l n l
n k * n nl n l n n n l n
n n n n n l ll n n
fl n n l n

Table 2. n n ll

u					u				
									
									
									
									

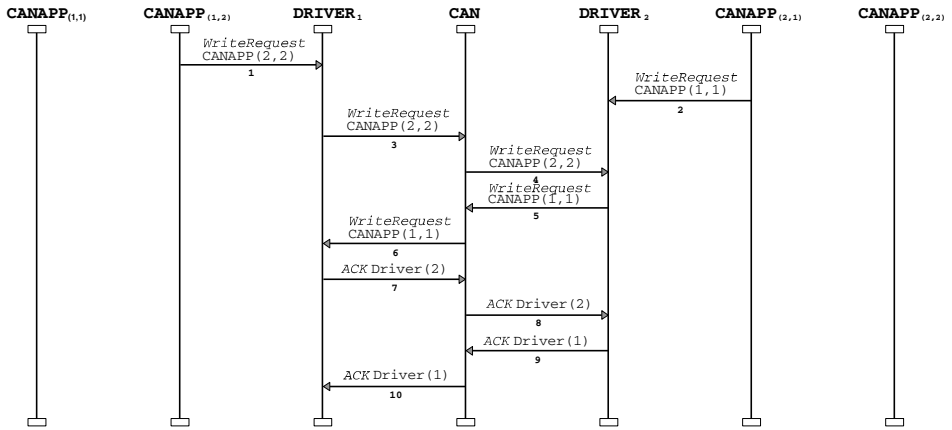


Fig. 7.

n ll n n n l l k l ll n n
 n n l k n n n n l
 l n n n n fl n n
 l k n l k n n n
 n n ListDeadMarkings ll n n n
 l n n n l k n k n n
 n n n n n ll n n
 (1,2) n WriteRequest (2,2) l
 n n WriteResponse (1,1) n (1,2)
 l k (2,1) n WriteRequest (1,1) (2,1)
 n (2,2) l k n l WriteResponse ll n fl
 l k n (1,1) n (2,2) n
 WriteRequests n l k
 ll l k n n ll k n l k
 l n l n fl l n l n n
 n l n l n n l n n
 (2,2) n l n l n
 l fl n l n n

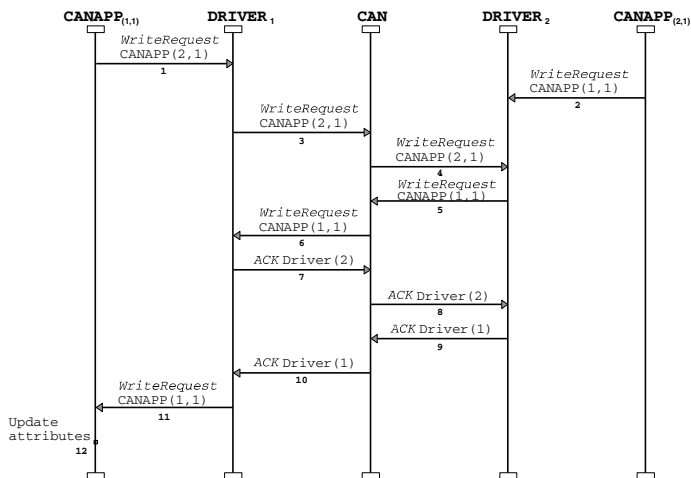


Fig. 8. 1 n 1 n n n

l k l n n n l ll
 l n n n n n l fl
 n l n n n n l
 n n n n k n l ll
 n n n l n l l n l k

 n l fl n n
 n ll n n n l l l
 n l n
 not l l l n
 n n n n k n n n
 n n fl n n
 l n ll n n
 n

 (1,1) n WriteRequest n (2,1)
 (1,1) n n n l k
 (2,1) n WriteRequest n (1,1)
 n n n kn l n n
 (1,1) WriteRequest (2,1)
 (1,1) n n l n WriteRe-
 quest (2,1) n

 l n l n n
 n n l k l l n
 n n n n n

u

n l l l n n n
n l l n n fl
n n n ll n n
l n n

6 Analysis of a Third Design Proposal

n n l k n n n n l
n l n ll l n n n l
l n l n n n n
l n n l l n n
n l n l n n
ll l n l n n l n n
l n l n fl
n n l n n
n n l l n
ll n n
n fl n n n n
l n n l n n
n fl
n n n l n l
n l n n n
n n n n n
ll n n n
n n n n ll n
l n l ll n l n
n n n l n n l n
n n

6.1 Symmetry Specification

n n fl *state spaces with permu-*
tation symmetries n l l
n n l
n n l n n
n n n n *symmetry groups* n
atomic colour sets l n l l
n n l ll n n

u

n l n n n l l n

l n n k n n

l n n n l

k n n n l n

n l k n k n l n n n n n

ll l n n n n n k n n

n l k ll

k n n l fl

6.3 Analysis Results

n l n n l n

l n n n kn l n n

n l l n l n nn

l kn l n l l

n n kn n

n n Indication n n n

nn n n n n

nl Indication n n n

n n n l mode n

accept

AG Request, $\langle canapp$ $(i,j) \rangle \Rightarrow$

A \neg Indication, $\langle canapp$ $(i,j), mode$ $accept \rangle$ U

Confirm, $\langle canapp$ $(i,j) \rangle$

n l l

n n l n ll n

n n n

l l l n n

n n n n k n

(i,j) n n ll

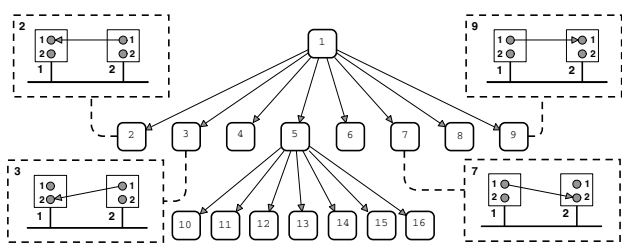


Fig. 9. n l n ll

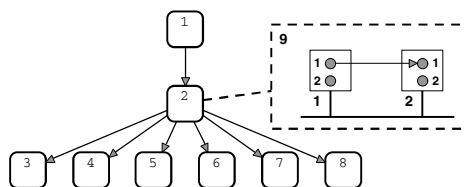


Fig. 10. n l n n n

[illegible]

Table 3. n n ll n n n

nn
n nn l n l n l n
n n n n n n
nn l l n n n n n n
l l n n ll ll
n l l n n n l l n ll
l n n n k n n
n n n n n n l n Sym
n n n n n n n n
n k n n n n
n n n l l n n
fl l n n l ll
l l n

u
 n n n n n n ll
 n nl n n n n
 l n k n n n l n n n
 n n n n l n n n
 n n n l n k n n ll n
 n n n l n n n l
 n n n n l n n l
 n n n n l k n *home marking* n n
 l k n l l n n l k n
 n n fl n n
 n n n nn n n l
 n n l n n n
 n n n n l n n
 n n n *live* n ll n n
 l l l n n n l n n n

7 Conclusions

n n n l n n
 n n l fl n n DANFOSS ll n
 n l n l n l
 l l l fl n n n
 fl n n n n
 n n l ll n n n n
 n n n l l n
 n nl l n l n n
 l n n n n l n n
 ll n n l n
 n fl n n n n
 n n n n ll n l n
 k n n l n n n l n
 n n l n l n n
 l n n n n n
 n n n n

Acknowledgements. I thank the DANFOSS INSTRUMENTATION
for the financial support.

References

u

u

u

.....

.....

..... ü u

u k

u

.....

.....

u

u

.....

u

uu

u

u

.....

j
i i i
- 00
k.h.mortensen@daimi.au.dk

$$\frac{y}{2} \leq y \leq y + m$$

y m
m y m y y
m

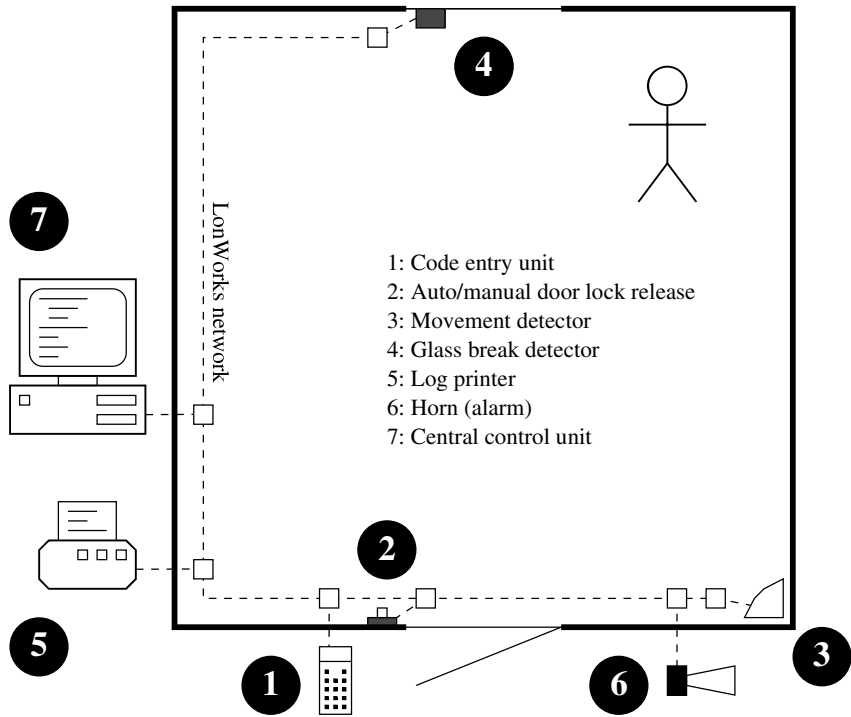


Fig. 1.

y m

k

2.1 Access Control System Scenario

m k m
y y y m m

Basic System Functionality:

y m k y y
m m y y m
y k m
m 2 y m
m y m

0

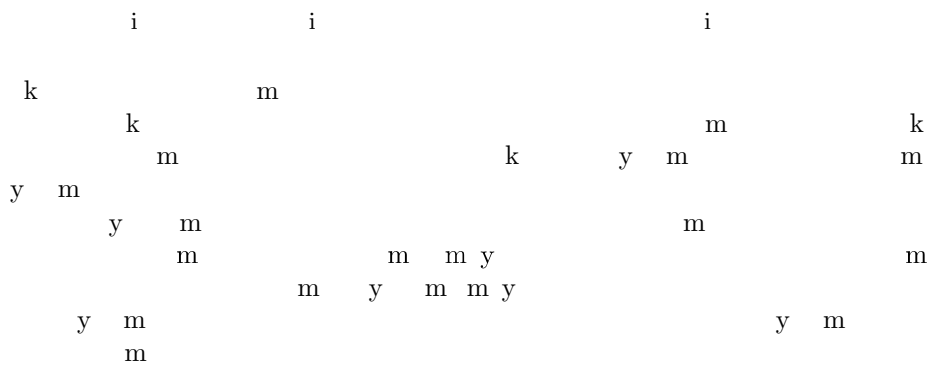
m y m m y m 2
k m
m m m
y m y

LonWorks, LonTalk, and Neurons:

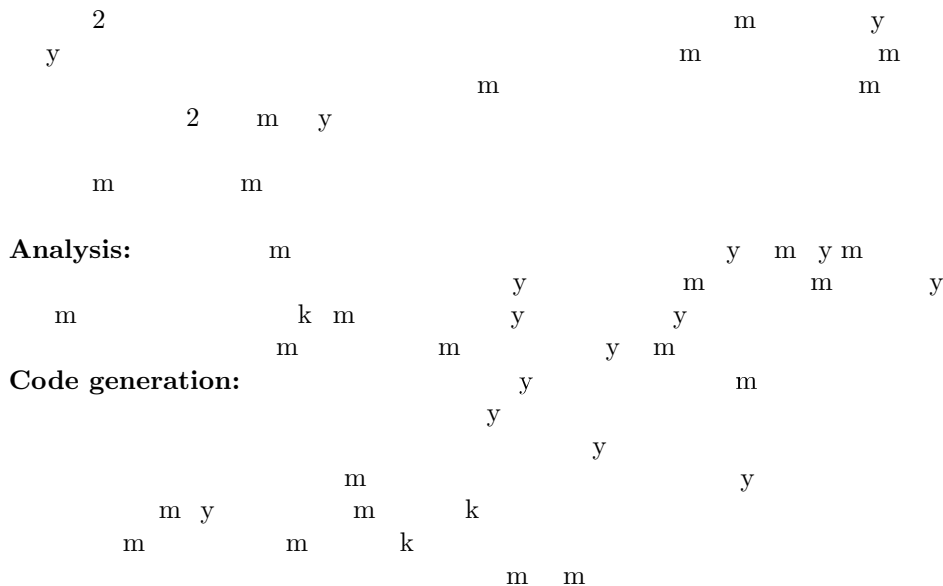
y m k m
y k ¹ k
k y y m k m
m k k k
network variables k *input* k
output k k
m k k *Neuron chip*
k m
k k m
k y m m y m
m k y m
m k y m m y k
y m k m y y k m k m
m m y m m k
y m m m

Characteristics of Dalcotech's Environment:

m y j
m m k m
k y m y m
k m k y
m k y m y
m m m m



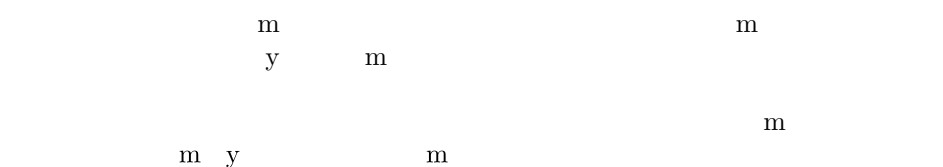
2.2 Automatic Code Generation for Access Control Systems



Specialising to environment:

Specialising of ML dialect:

Generation of system to central controller:



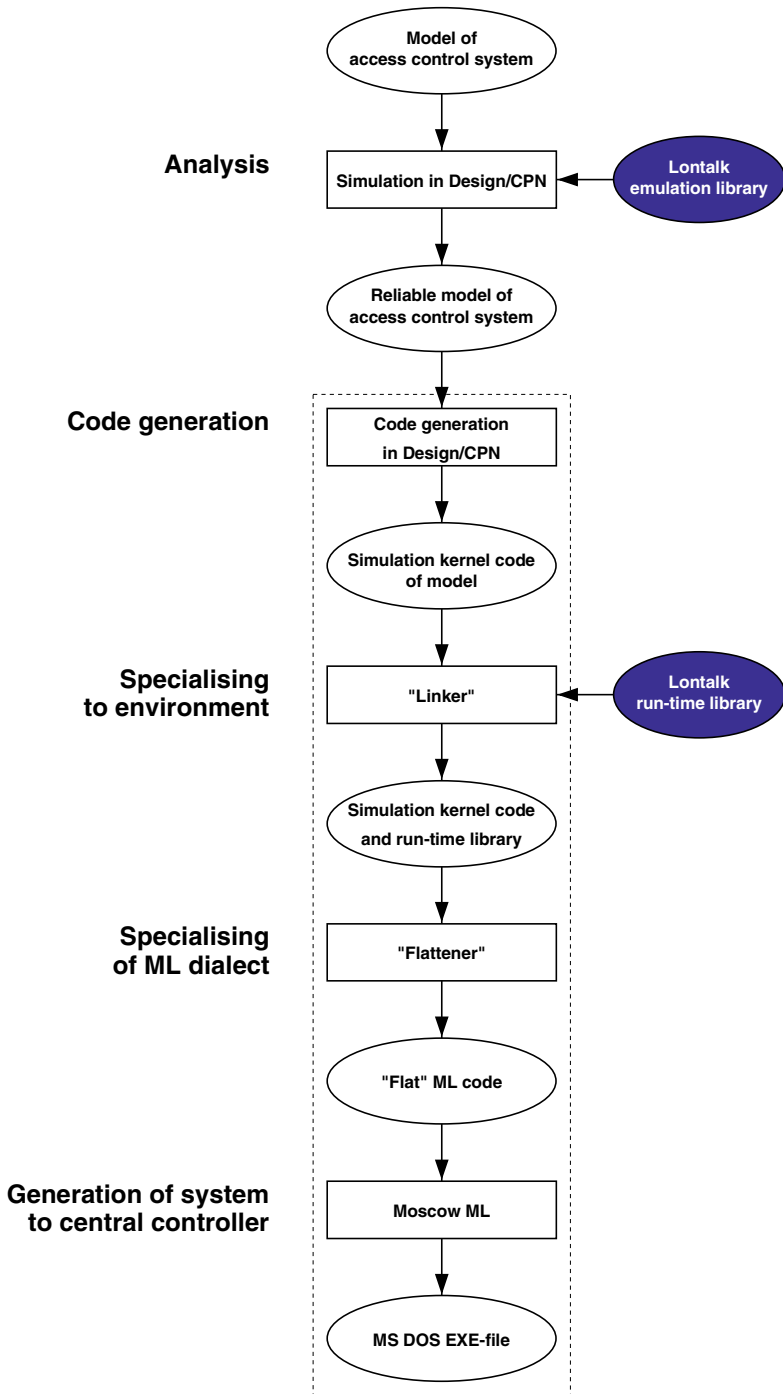


Fig. 2.

m

y

3 CPN Model of Access Control System

3.1 Selected Parts of Model

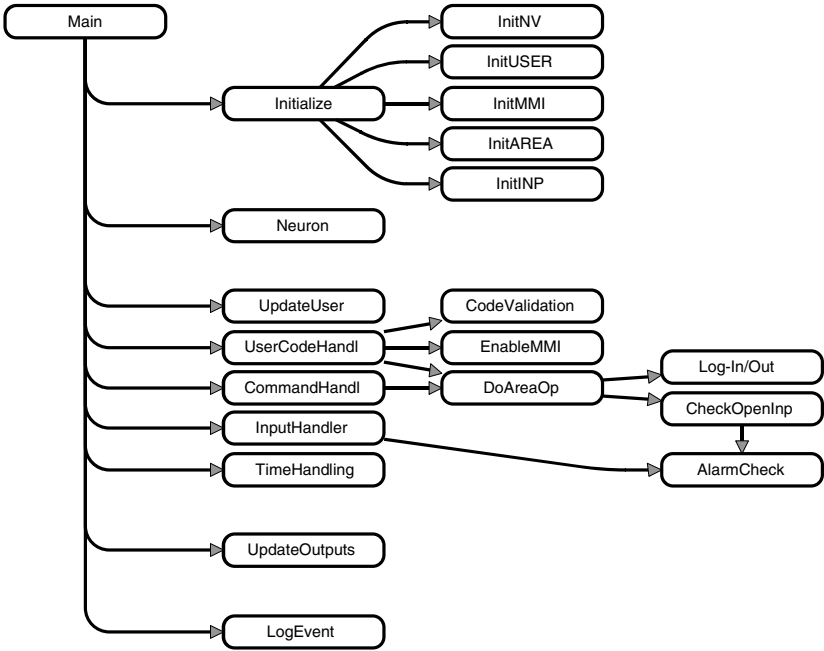


Fig. 3.

Hierarchical Structure:

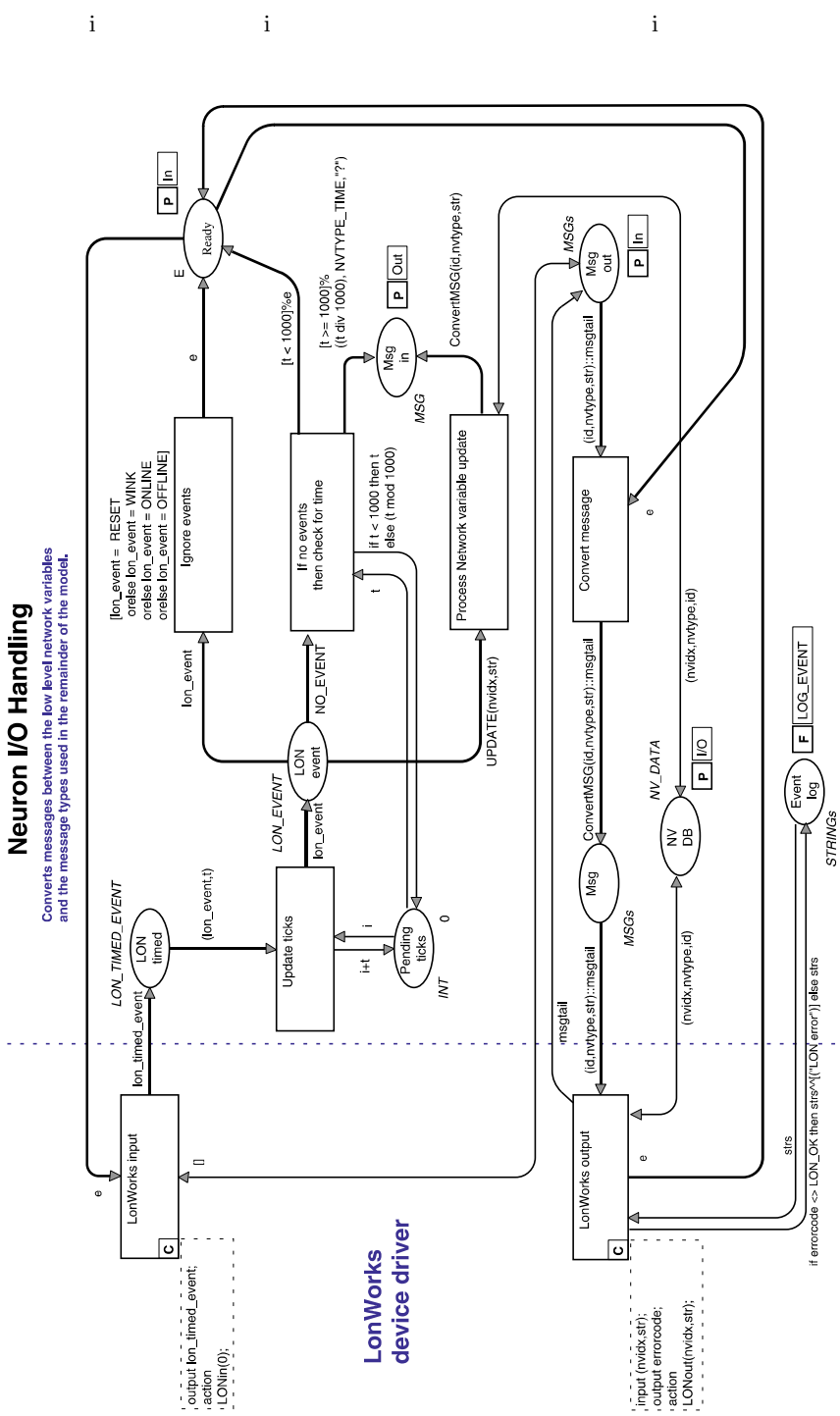


Fig. 4.

```

      k          m          m          m          LonWorks input
      m          m          m          m          y
LonWorks output
m          y          y          m k          y          k
      y          LONin          LONout          m          k
LONin          k          m          m          k
      m          NO_EVENT
          k m          m
      m          k m          m          NO_EVENT
          m          k          m          m          m
      m          m m          m          m          m
          m          k m
      LON_TIMED_EVENT
          m          m          m          m
m          y          y m          m          m
          y m k          m          m          m
          m          m          y m          y m
      k          m          y          m          y
      m          y          m          m
          m          y          m
      k          k          m
m

```

Event Handling:

```

      y          m y y m
      y m          m
      y

```

```
colorset MSG = product
```

```
  INT          (* Id *)
```

```
  * NV_TYPE    (* Network variable type *)
```

```
  * NV_VAL;    (* Data *)
```

```
colorset MSGs = list MSG;
```

```
      y m
```

```
          m          m          k
```

```
Neuron Input/Output          k m
```

```
  Msg in          y
```

```
          User changed User code handling Command
```

```
handling Input change handling Time handling          y
```

```
          y          k          y m
```

```
      k Update outputs          m
```

```
      k          y
```

```
m          k          m
```

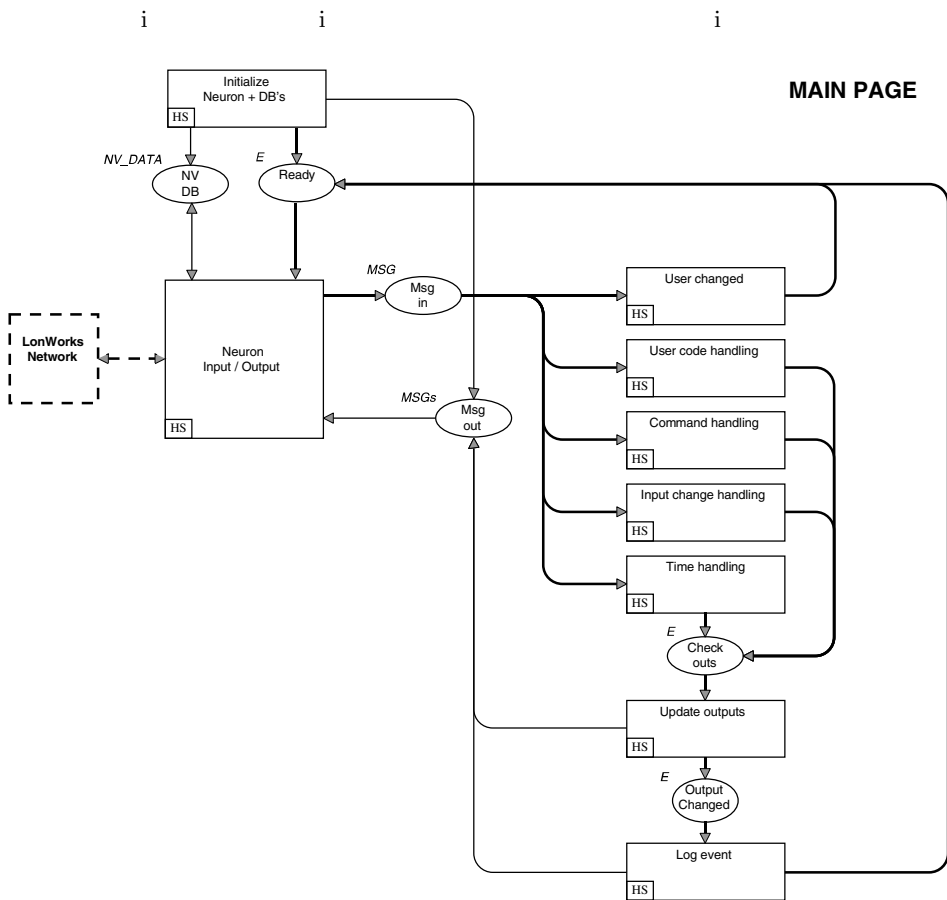


Fig. 5.

m m y

m y m m y

y m m k y

m j y m m m

m y y m k m m m

m y m k m m m y

m k m m y m

3.2 Benefits of Model Architecture

m m y

m y m
 m y m m y m
 y m
Extensible: m
 m k
 m y y k
 y m y j m y m
 y k y m y
 y

Reusable: m
 m k y m y m
 m y m
 y m m m
 m k m
 m k m y
 m y m m m m

Adaptable: m
 k k k y m m
 y y y m m m 20
 y m y m m m
 z

3.3 Analysis of Access Control System

y y m fly y m m
 m y k m m
 y j k m y
 m

```

i          i          i
          m          y
k      k      k m      y
      m      y      k m      m      k m
LONin      y      k m      y      m
      y      k m

3.000
UPDATE(18,"\001")
2.000
UPDATE(18,"\000")

          y          k      m
        y m      m      m      2
          k      z      m      m      y      m
y m k m y      m
      m      m

```

3.4 Code Generation of Access Control System

```

m          m          m y
          m
        m      2      y
y      m m

Techniques and Tools:          m          m
          k      y          y      y
k          y          m
use "_LonTalk.sml"
          k      m      y
m      m      m      m      y
y /tmp/access-ctrl.sml      m
00      2 0 k
          cpnsim2mosml.pl
k      m      y      m
      y m      fl

linux% cpnsim2mosml.pl /tmp/access-ctrl.sml /tmp/access-ctrl.mml
          access-ctrl.mml      2 000      20 k
m      access-ctrl.mml      m
m      y      m

msdos% mosmlc access-ctrl.mml

mm          mosmlout.exe
          m      k      k

```

z 20 k m y
z 0 k
00 2000 y k 0 20 m 200 z m
m

3.5 Performance of Access Control System

m y m m y m
m m y m
m y m 0 z m
m y m y m y y m m
m y m 0 2 0 m
k y m m y
m y

3.6 Main Benefits of Method for Dalcotech

m j k m m y m
m y m
mm y m same
y m
- y m m
- y m y m m m y
m y y m

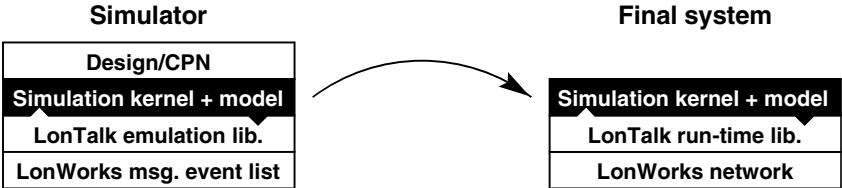


Fig. 6. both m
y m m m

y m m m y m m m

i i i
 — m m m m y ; m k
 20 m
 — m m m y
 m y
 m m y m y
 m m y m
 m m
 m m m m y m
 m m m
 m m
 m m

4 General Method Developed in AC/DC Project

m m y
 y m m
 m j
 m mm
 m 2 m
 m y m k y m
 y m y
 m k y
 m y
 m m
 m m
 m y m
 m m
 m m m
 m y m
 2 m m
 mm m z m
 m m k
 m k m k

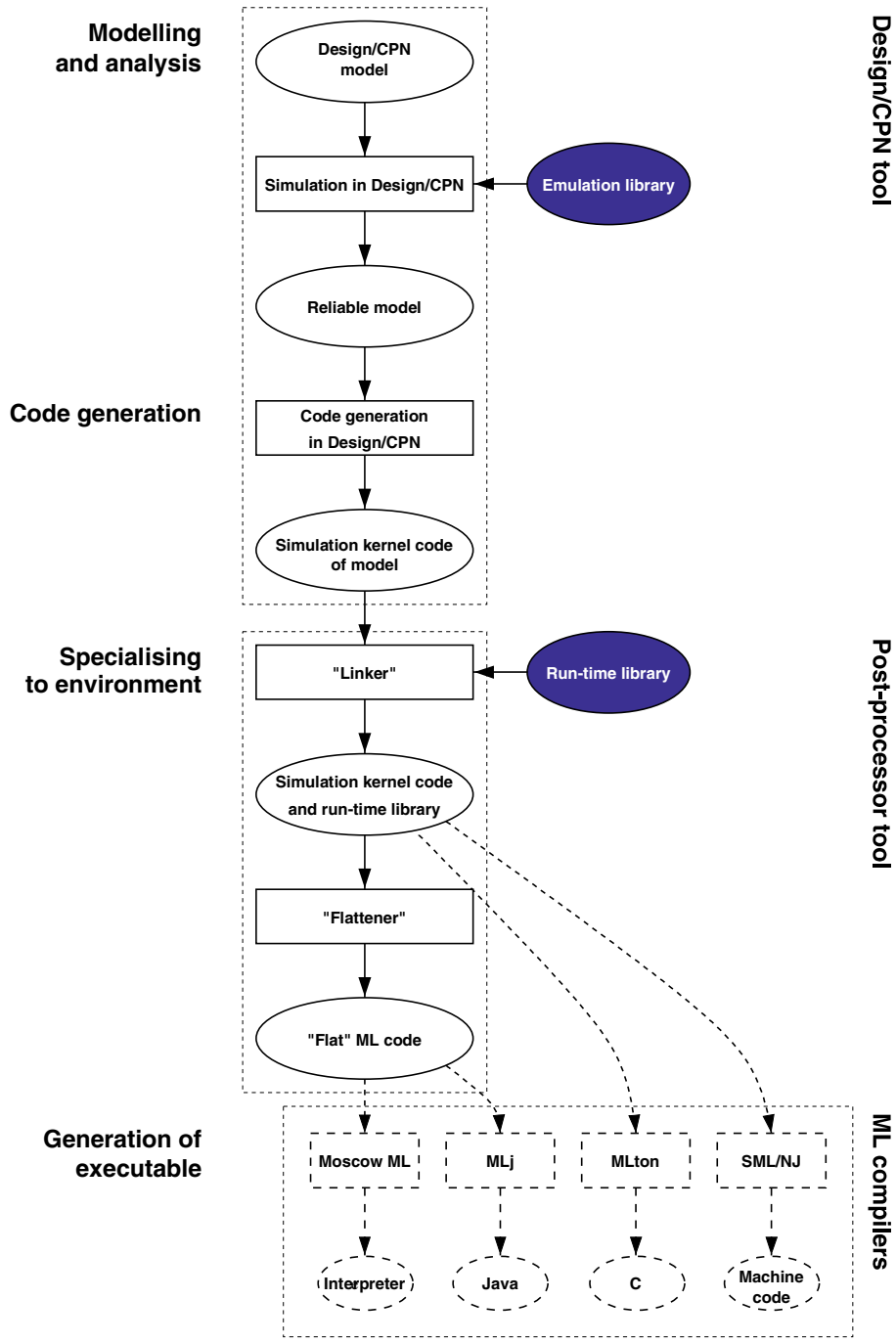


Fig. 7.

j

m

i	i	i	
m		m	z
		0	
	m	j	y
m	m	j	
j	m		m
	m	y	m
			m

5 Related Work

m	0			y
20	y	m	m	
y	y	m	y	m
y	k		m	m
		m	j	y
m	m			
	y			
m	m			

CPN-AMI:

	m	m	m	k
	m			m
y	m			
		m		
	m			
			m	
j	m	k		m
		2	m	
m				
	m		m	

PEP:

	m	y	k	
k	m	y	k	
				$B\;PN^2$
				m

m y m
 m m
 m y m mm j y m
 m
 m m m m
 m m y k
 y m y m y
 m m y m k
 j 2

6 Conclusion

m y m y m y m y
 m m y m m m y y
 y m m m m m y y
 m y m m m y
 m m j m y m y
 y y m m y y m
 m m m m y m
 m y m m y m
 m k y m m
 y m m y
 y j m y
 y y m m y m
 m m y m
 m m m k
 m m k

Future Work

y y m y m m

i i i
m m m m
m m m m y
m k m k y m y y
m y m m y m m
2 m m m
m y m m m y m
m m m m y m
m m y m m
m m y m
k m m
m 2 y y
m y k m m y m
m m m

Acknowledgements

j y y
2
m y

References

1. i i i *Workshop on Object-oriented Programming and Models of Concurrency of the 16th International Conference on Application and Theory of Petri Nets* i
2. i i z i *Workshop on Object-oriented Programming and Models of Concurrency of the 17th International Conference on Application and Theory of Petri Nets* 6
3. - i i -
4. i i i i 0
5. 6 i i i 0
6. *Tool Presentations of ATPN'97 (Application and Theory of Petri Nets)*
7. i i i i i -
8. 6 i i i

		<i>Coloured Petri Nets — Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts</i>	i	i	i
		i	i	-	
			i	<i>CIT NYT</i>	i i
		i	i		
				..	i
		<i>Oriented Environments: The Mjølner Approach</i>	i	-	<i>Object-</i>
0		i	i		i i i
		i		<i>International Journal on Software Tools for Technology Transfer</i>	(
		-		-	i
		i			i
			i	i	i
					i
		-			
		i			<i>The Definition of Standard ML</i>
0		i	i		
		i			i
		i		i	-
		i			06
		-			
		<i>Design/CPN Automatic Code Generation User's Guide</i>	i	i	
6		-		<i>Design/CPN LonTalk Library User's Guide</i>	i i
					i
		<i>ML for the Working Programmer</i>		i	i i
		i i	6		
			i	i i	i
		i	i i	i i i	
		<i>17th International Conference on Application and Theory of Petri Nets 1996</i>	0		<i>Lecture Notes</i>
		<i>in Computer Science</i>	00	6	i -
		<i>Moscow ML owner's manual</i>		i	i
0		i	~		
		..		i	i -
0		i -	i i i	-	i i
		i			..
			i	i	i
		i	i	i	i
			i	i	i
		i			
				06	i
		i i			
		i		i	-
		i	i		i
6		i		i	i
		i i			
			i		
		i	i	i	
0		i		i	i i
			i		i
		i		i	i
				i i	i
				i i	i

Pre- and Post-agglomerations for *LTL* Model Checking

1

2

1

2

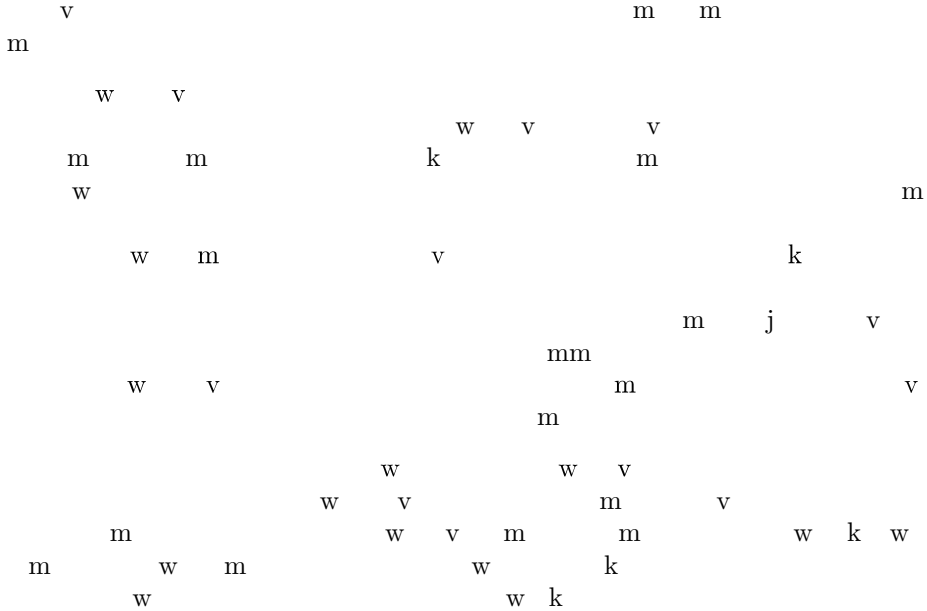
9

Abstract.

q

1 Introduction

k m k k m v m
m k m v k
m m v m w k
k m m v k
m k m m k w k w m
m m m w v v
m v k
v m v m v
v m m v m
v v w m
w m m



2 Common Definitions and Properties



2.1 Definition of Petri Nets

Definition 1. A Petri net [4] is 4-tuple $\langle P, T, \text{pre}, \text{post} \rangle$ where: P is the set of places, T is the set of transitions (T^- (resp. T^+) is the backward (resp. forward) incidence application from T to P), pre (resp. post) is the backward (resp. forward) incidence application from T to P .

A marked Petri nets is a couple (N, M_0) where N is a Petri net and M_0 is the initial marking.

Definition 2. Let (N, M_0) be a Petri net and M_0 its initial marking. A vector of \mathbb{N}^P is called a marking of N ; $M(p)$ denotes the number of tokens contained in place p . A transition $t \in T$ is fireable at a marking M if and only if: $M - \text{pre}(t) \geq 0$. The marking M' obtained by the firing of t is defined by: $M'(p) = M(p) - \text{pre}(t)(p) + \text{post}(t)(p)$. One notes, that means that t is fireable at M and reaches M' . By extension, a marking M' is reachable from a marking M if there exist a sequence t_0, t_1, \dots, t_n and a set of markings M_1, \dots, M_n such that $M \xrightarrow{t_0} M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} \dots \xrightarrow{t_n} M_n$. The set of reachable markings from M_0 is denoted by $R(N, M_0)$. A maximal firing sequence is either an infinite fireable sequence or a finite fireable sequence leading to a terminal marking (i.e. a marking from which none transition is fireable).

Notation.

$$\begin{array}{ccccccccccccccc} & & \omega & & & & m & & & & 0 & & & & \\ & m & 0 & & \omega & & 0 & & m & 0 & & 0 & & max & 0 \\ & & m & m & & & & & & & & & & & \end{array}$$

Notation.

$$\begin{array}{ccccccc} & & & \omega & & & \\ - & & & & & & \\ - & & & & & t & m \\ - & T' & & \sum_r T' & r & & \\ - & T' & & & & m & \\ - & i & & i^{th} & & i & \\ - & & & & & & \end{array}$$

2.2 The Pre- and the Post-agglomerations

$$\begin{array}{ccccccccccccccc} v & w & j & & & & m & & & & & & & \\ & m & & & & & & & & & & & & \\ & w & & & & & mm & & w & & & & & \\ & & & & v & & & & & & & & & \\ & & & m & & & & & & m & & & & \\ & & & & & w & & & m & & & & & \\ & & & & & & & & & & & & & \\ & m & & & m & & & & & & & & & \end{array}$$

Definition 3 (Agglomeration scheme). Let \mathcal{N}_0 be a marked Petri net. Two disjoint subsets of transitions T_1 and T_2 and a place p satisfy a structural scheme of agglomeration if and only if:

$$\begin{array}{ccccccc} - & & & and & & & \\ - & 0 & & & + & & 1 \\ - & & & & & & \\ & & m & & & & \\ & w & & & m & & m \\ w & & m & & m & & \end{array}$$

1

Definition 6 (Net reduced by a pre or a post-agglomeration).

The reduced net r_{0r} obtained by the application of a pre-agglomeration (resp. of a post-agglomeration) on r and on the net 0 is defined by :

$$\begin{aligned} & - \quad r \quad , \quad r \quad 0 \quad , \text{ we note } \quad \text{the transition} \\ & - \quad 0r \quad 0 \\ & - \quad r, \quad r^+ \quad r^- \\ & - \quad r \quad , \quad r^+ \quad + \quad \text{and } \quad r^- \\ & - \quad r^+ \quad + \quad r^- \quad (resp. \quad r^+ \quad + \quad r^-) \text{ and } \\ & \quad r^- \quad (resp. \quad r^- \quad - \quad r^-) \end{aligned}$$

Remark 1.

$$\begin{array}{ccccc} & m & & v & \\ & w & v & k & \\ w & & v & m & k \\ & & & v & \end{array}$$

Fundamental Properties Equivalence

$$\begin{array}{ccc} v & m & \\ v & m & \end{array}$$

Proposition 1 (Berthelot 85 [1]). Let \mathcal{P} be a main property (liveness, boundedness, home state existence,). Then :

$$0 \text{ satisfies } \mathcal{P} \text{ iff } r_{0r} \text{ satisfies } \mathcal{P}$$

2.3 Linear-Time Temporal Logic

$$\begin{array}{ccccccc} m & m & LTL & & m & & \\ & m & & & & & \\ & & m & & m & Verif & m \\ AP & & m & & LTL & v & w \\ v & m & LTL & m & LTL & m & \\ & & LTL & m & w & & \\ & & & & & 0 & 1 & 2 & v \\ & & & & & LTL & m & & \\ w & & w & i & & i & & & \\ & v & w & & & w & LTL & m & \\ i & 0 & & & & & & & \\ & & & & & & & & \\ 1 & & & & & & & & \\ i & & i & & i & j & & & \end{array}$$

$$LTL \quad m$$

$$\begin{array}{ccccccc} & & w & & & & m \\ & & & m & w & & \\ & & & w & m & & \\ w & & 0 & w & m & & \\ & & & & & & \end{array} \quad \begin{array}{c} Verif \\ m \end{array} \quad \begin{array}{c} v \\ w \end{array} \quad \begin{array}{c} m \\ w \end{array}$$

State-Based Linear Temporal Logic

$$\begin{array}{ccccccc} m & & & & m & k & \\ & & 0 & & & & \\ & m & m & & 0 & AP & 0 \\ & & & & & & 0 \quad 0 \\ 1 \quad 1 & & & & 0 & 1 & \\ & v & w & m & w & 0 \quad 0 & 1 \quad 1 \\ \\ - & & & & s & & \\ - & & & & & & \\ & w & & & & m & \\ - & & m & & & m & k \\ - & AP & & & & & \\ - & & & i & & & \\ - & & 0 & max & 0 & & \\ & & & m & w & & \\ LTL & m & & & & & \\ m & & w & & & & \end{array}$$

Proposition 2 (Stuttering property [3]).

Let φ be a state-based LTL formula. Let $w = i_0 i_1 i_2 \dots$ be an infinite word over Σ . Let $w' = i'_0 i'_1 i'_2 \dots$ be the infinite extracted word of w defined by:

$$\begin{array}{l} - \quad i_k \quad i_{k+1} \\ - \quad i \quad i_k \quad i \quad i_{k+1} \quad i \quad i_k \\ - \quad i_k \quad i_{k+1} \quad i \quad i_k \quad i \quad i_k \end{array}$$

Then $\varphi(w) \iff \varphi(w')$.

$$\begin{array}{ccccccc} m & & w & & v & & \\ & & & m & & & \\ & & 0 & & & & \\ & & & & w & & \end{array}$$

Corollary 1. *Let φ be a state-based LTL formula. Let ω be a trace such that $\omega \models \varphi$. Then $\omega \models \text{Obs}(\varphi)$.*

m m m v
 m m k w v
 m
 j m m

Action-Based Linear Temporal Logic

Action-Based Linear Temporal Logic

m w k w
 m m m
 w v
 m w 0 0 1 1
 $-$ w
 $-$
 w m
 $-$
 $-$
 $-$
 $-$ i AP^∞ max 0
 0 v v m

Proposition 3. *Let φ be a action-based LTL formula. Let $\mathcal{M} \models \varphi$ such that $\text{Obs}(AP) \subseteq \text{Obs}(AP)$ then*

Proof. v

3 Agglomerations and *LTL* Model Checking

m w m k 0 w
0
m w 0
m m w
v m v
k m m v

w v LTL m w

v

m r r

Definition 8. Given a logic (ϕ, ψ) and (ϕ, ψ) and a marked Petri net (N, M_0) the mappings α_r and β_r associated to the reduced net (N_r, M_{0r}) are defined by:

- If (N_r, M_{0r}) is reduced by a pre-agglomeration then α_r is the mapping defined from α_r to β_r by: $\alpha_r(s) = \beta_r(s)$ if $s \in M_0$ and $\alpha_r(s) = \beta_r(s)$ if $s \in M_{0r}$.
- If (N_r, M_{0r}) is reduced by a post-agglomeration then α_r is the mapping defined from α_r to β_r by: $\alpha_r(s) = \beta_r(s)$ if $s \in M_0$ and $\alpha_r(s) = \beta_r(s)$ if $s \in M_{0r}$.
- For both pre and post-agglomeration, β_r is the mapping defined from α_r to β_r by: $\beta_r(s) = \alpha_r(s)$ if $s \in M_0$ and $\beta_r(s) = \alpha_r(s)$ if $s \in M_{0r}$. The mapping β_r is defined from α_r as in section 2.3.

The set β_r is defined by extension of α_r or β_r .

Theorem 1 (Restricted equivalence).

Let ϕ be state-based LTL formula (resp. an action-based LTL formula). Let α_r be a maximal firing sequence of the reduced net obtained by a pre or a post-agglomeration.

If α_r in the case of the pre-agglomeration or β_r in the case of the post-agglomeration, then

$\alpha_r \sim \beta_r$ (resp. $\alpha_r \sim \beta_r$)

Proof.

w k

– m m m

v

r 0 r 1 r 2 m w v

0 1 1 2 w v 1

1 r 1 1 m w

r r r r

– m

r r r r

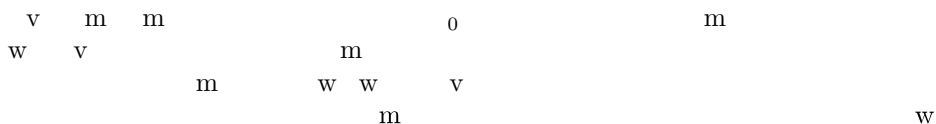
m m

m v

m w v m

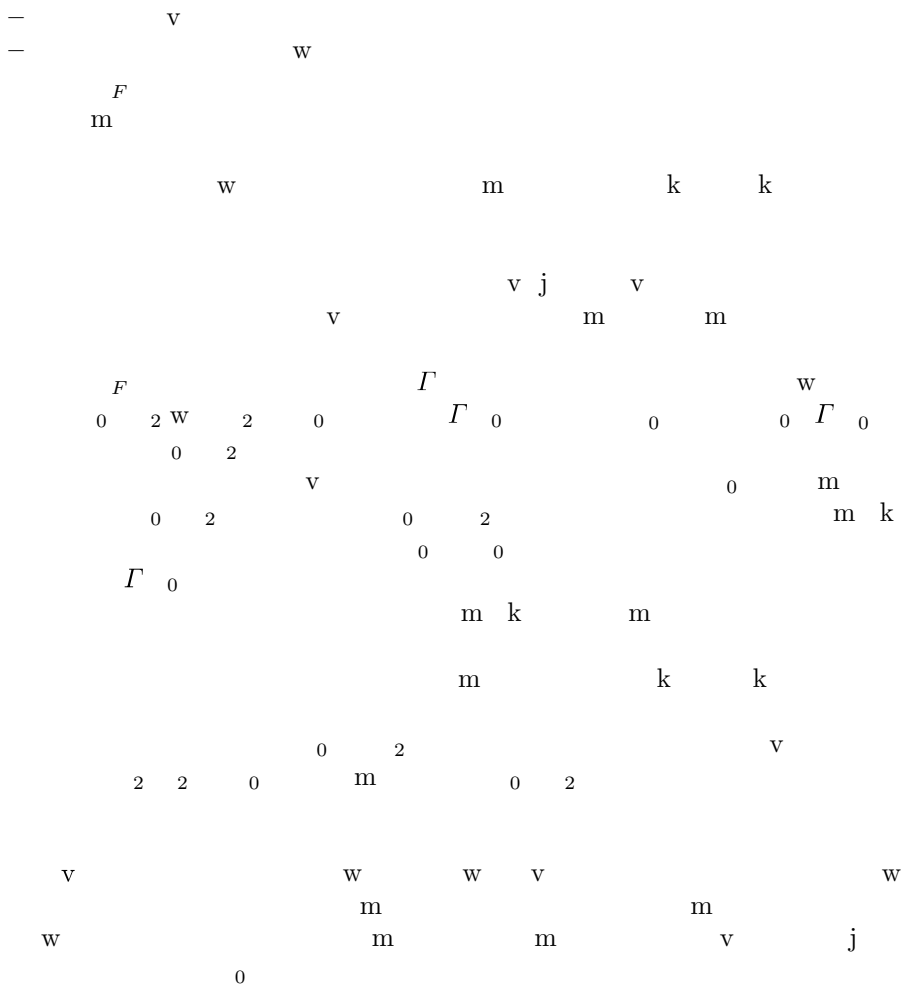
m m v

3.3 The Pre-agglomeration and *LTL* Model Checking

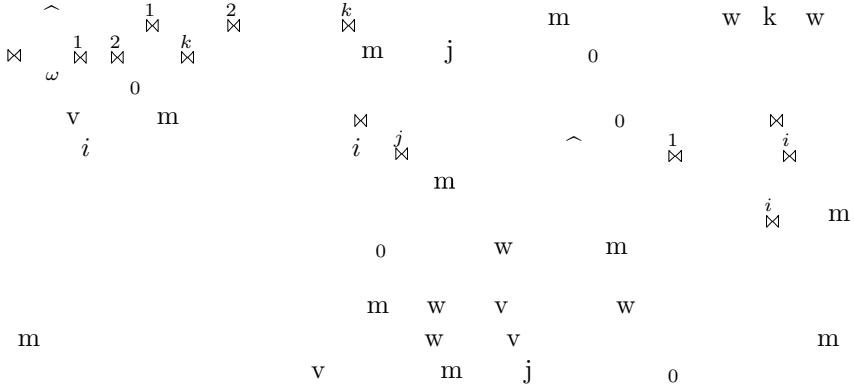


Proposition 8. *Let μ be a reachable marking and $\gamma \in \mathcal{G}$. If $\mu \xrightarrow{\gamma} \mu'$, $\Gamma(\mu) \subseteq \Gamma(\mu')$ then $\mu' \in \mathcal{R}(\mu)$.*

Proof. m k

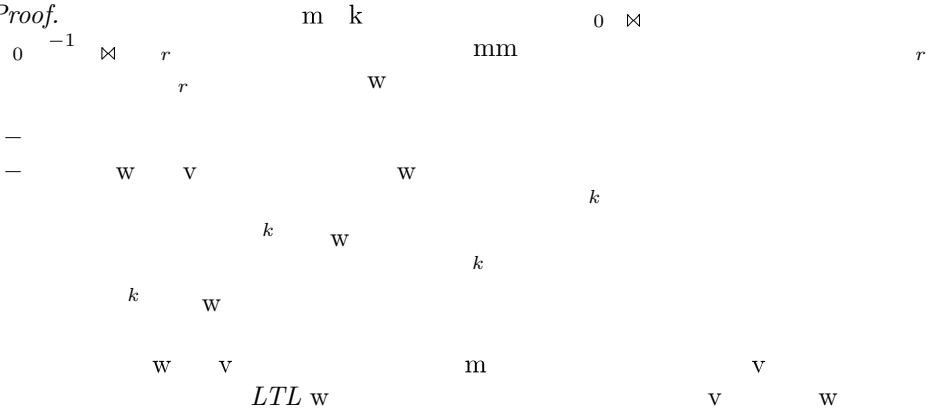


Proposition 9. *Let μ_0 be a reachable marking from μ_0 and σ be a sequence from μ_0 to a marking μ such that $\mu \in \Gamma$.*



Lemma 2. Let σ_0 be a sequence leading to a terminal state. We note $\hat{\sigma}_k$ a reordering of σ_0 . Then $\sigma_0^{-1} \hat{\sigma}_k$ is a sequence of the reduced net leading to a terminal state (and by construction, has the same projection over σ_0 than σ_0).

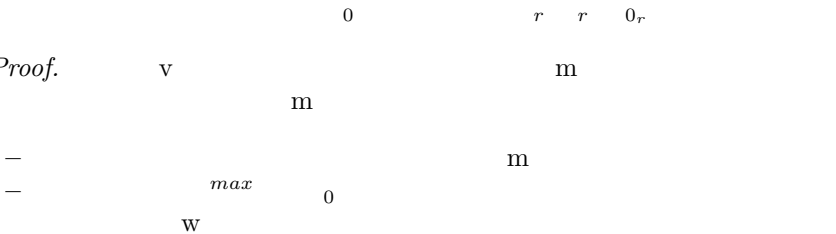
Proof.

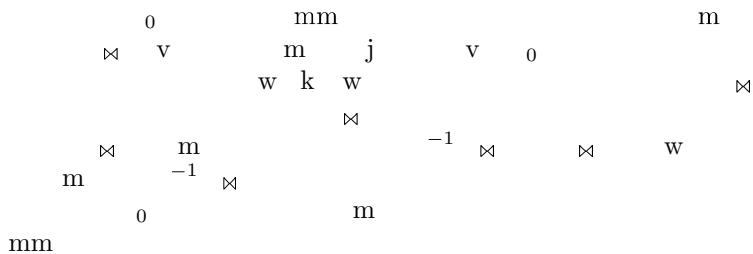


Theorem 2 (Pre-agglomeration and model checking).

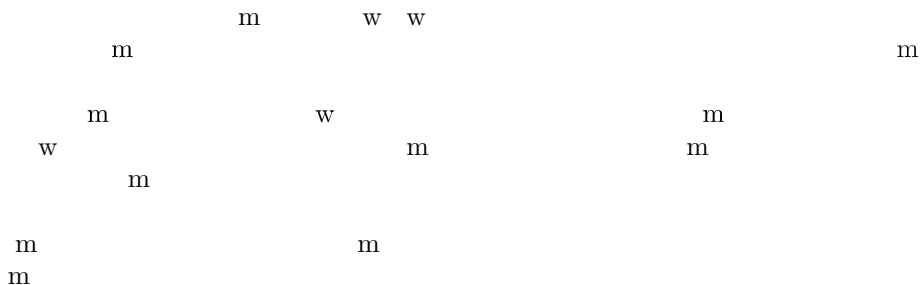
Let ϕ be a state-based LTL (resp. action-based LTL) formula. Let σ_0 be a net and σ_r σ_{0_r} be the corresponding reduced net by a pre-agglomeration w.r.t. ϕ and σ_0 . If $\sigma_0 \models \phi$ then $\sigma_r \models \phi$.

Proof.





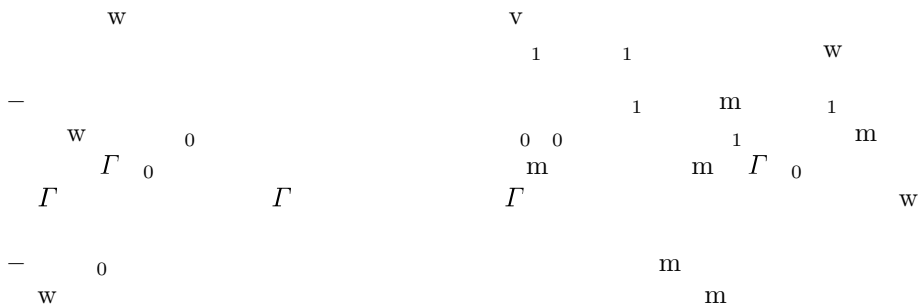
3.4 The Post-agglomeration and *LTL* Model Checking



Proposition 11. *Let μ be a reachable marking and let σ be a sequence of transitions such that $\sigma \in \Gamma$. Then $\mu \xrightarrow{\sigma} \mu'$, where $\mu' = \mu - \sum_{t \in \sigma} \text{Pre}(t) + \sum_{t \in \sigma} \text{Post}(t)$.*

and

Proof.



Corollary 4. *Let μ be a reachable marking and let σ be a sequence of ω such that $\sigma \in \Gamma$. Then $\mu \xrightarrow{\sigma} \mu'$, $\mu' \in \mathcal{R}$, and $\mu' \models \varphi$.*



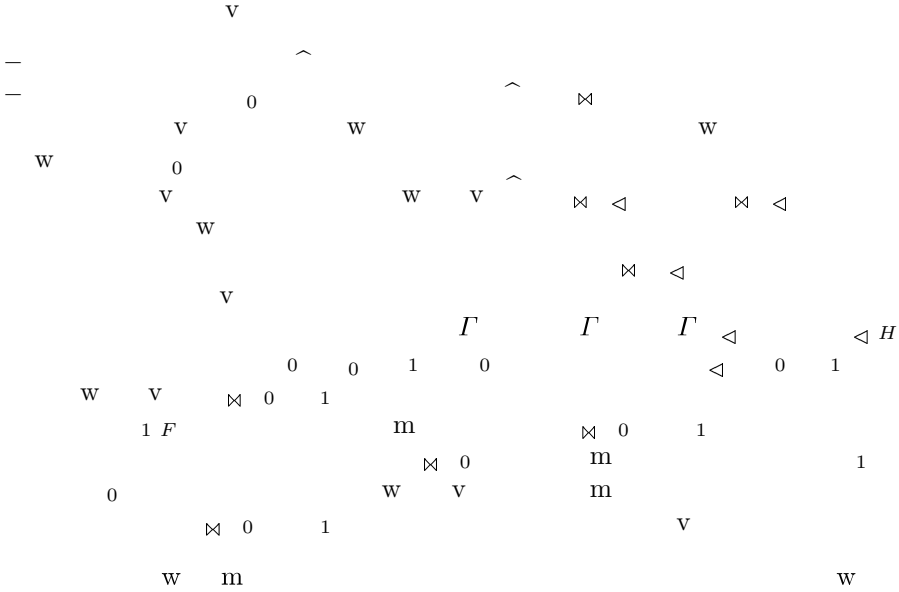
Proposition 12. *Let μ_0 be a reachable marking from μ_0 . Let σ be a sequence from μ_0 to a marking μ such that $\mu \in \Gamma$.*

There exists a permutation of $\hat{\mathcal{A}} \bowtie \triangleleft$, such that:

1. $\hat{\mathcal{F}} \triangleleft \mathcal{F}$ and $T_0 H \bowtie T_0 \hat{H}$
2. $\mathcal{F} \triangleleft \hat{\mathcal{F}}$ and $T_0 H \bowtie T_0 \hat{H}$
3. \bowtie is simulateable

Any sequence \hat{v} of v fulfilling the above requirements is called an *admissible sequence*.

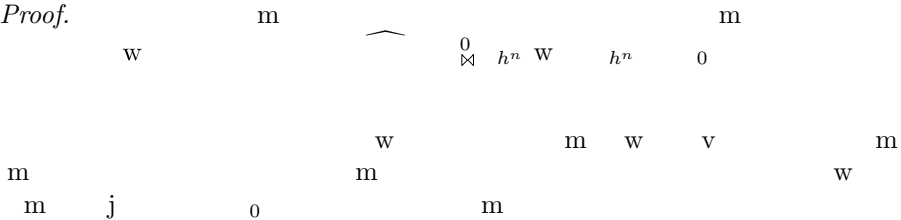
Proof.



Proposition 13. *Let $\mathcal{A} \in \mathcal{A}(\mathbb{R}^n)$ be a symmetric matrix. Then there exists a permutation $\sigma \in S_n$ such that $\mathcal{A}(\sigma(\mathbb{R}^n))$ is a direct sum of a symmetric positive semidefinite matrix and a symmetric negative semidefinite matrix.*

1. $T_0 \overset{H}{\underset{i}{\boxtimes}} 0$ an infinite series of simulateable sequences such that
- (a) $\overset{H}{\underset{i}{\boxtimes}} 0$ with h^n $\overset{0}{\underset{i}{\boxtimes}}$
- (b) $\overset{H}{\underset{i}{\boxtimes}} 0$ with h^n $\overset{0}{\underset{i}{\boxtimes}}$

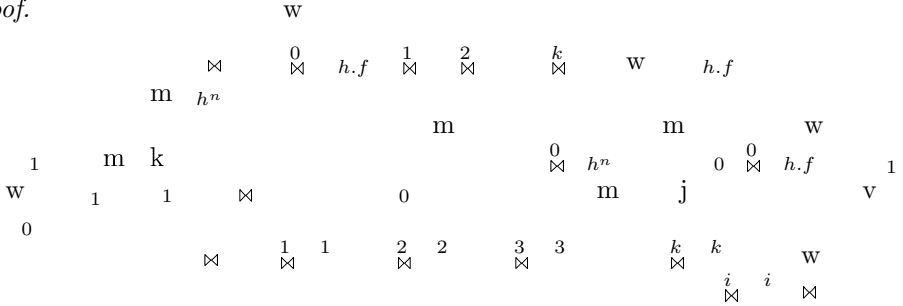
Proof.



Lemma 3. *Let \mathcal{A}_0 be a countable set of \mathcal{A} -names. Then there exists an infinite simulateable sequence $\langle \mathcal{A}_i \mid i \in \omega \rangle$ such that $\mathcal{A}_0 \subseteq \mathcal{A}_1$ and $\mathcal{A}_i \subseteq \mathcal{A}_{i+1}$ for all $i \in \omega$.*

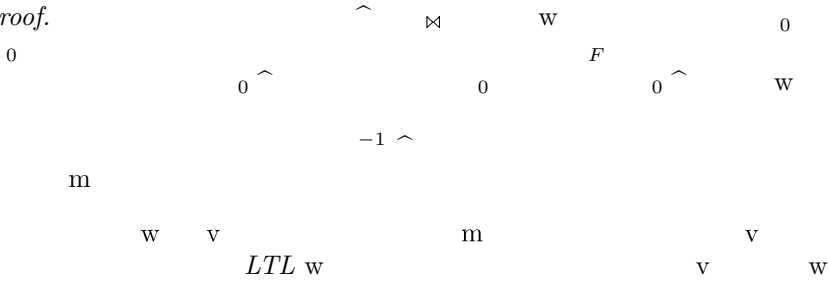
1. $\bowtie \quad \quad \quad 0$
2. $T_0 \ H \ \bowtie \quad \quad T_0 \ H$

Proof.



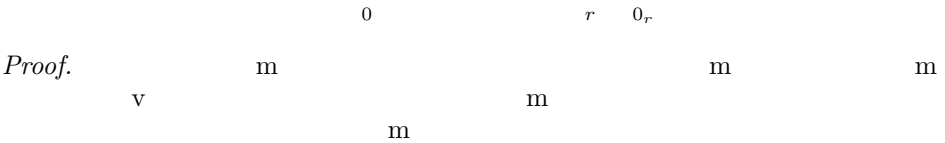
Lemma 4. *Let σ_0 be a sequence leading to a terminal state. Then σ_0 is balanced and $\sigma_0^{-1} \wedge$ is a sequence of the reduced net leading to a terminal state (and by construction, has the same projection over σ_0 than σ_0).*

Proof.

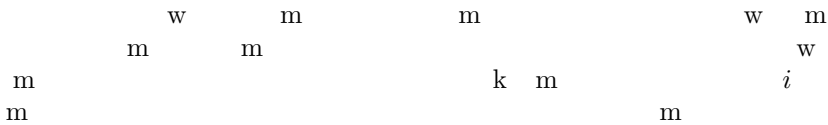


Theorem 3 (Post-agglomeration and model checking).

Let ϕ be a state-based (resp. action-based) formula. Let σ_0 be a system and σ_r, σ_{0_r} be the corresponding reduced net by a post-agglomeration w.r.t. ϕ and ϕ . If $\sigma_0 \models \phi$ then $\sigma_r \models \phi$ and $\sigma_{0_r} \models \phi$.



4 Experimentations



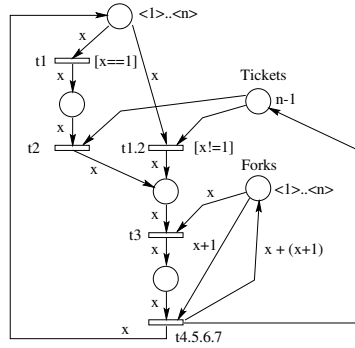
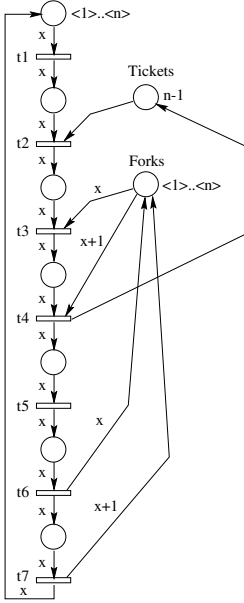
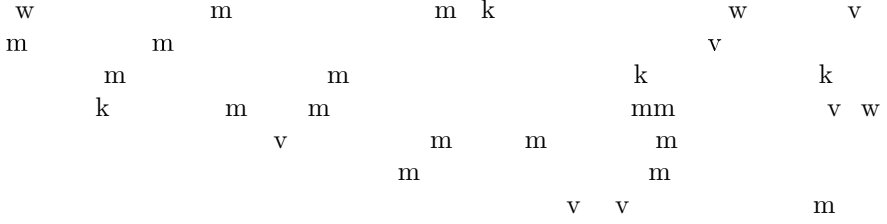
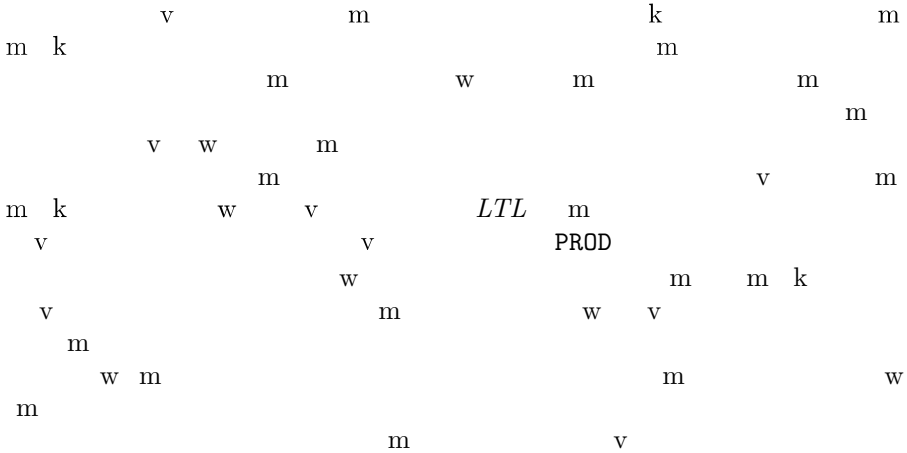


Fig. 1.



LTl
PROD

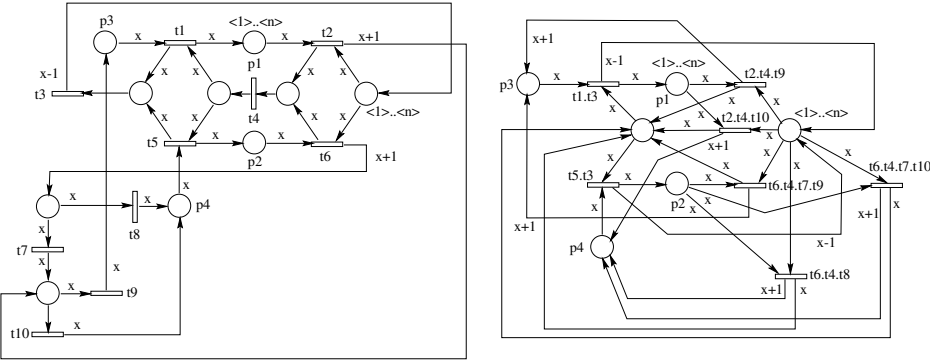
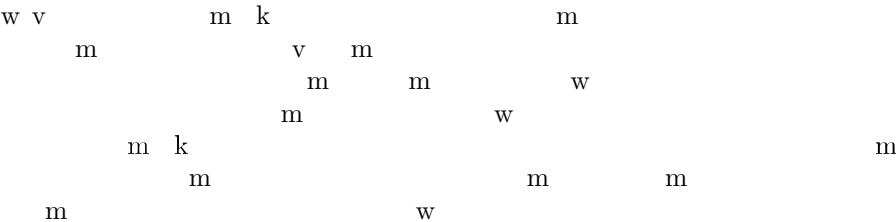
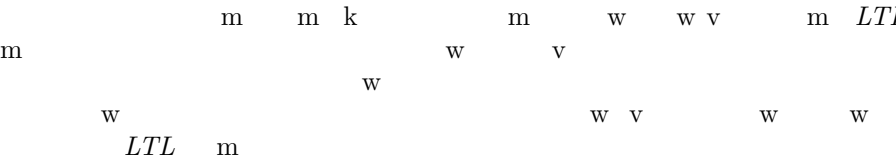


Fig. 2. m



	11	111	1				1	
1	1		9 9	1 9	1	9	1	1 9
	1	1	9	1 9	1 11		11	1
			1	1		9		

Table 1. m m k



—	1	1		
—	1	1	1	1

i i i i i v v m m v i
 i i i i v m v m w
 m k 1 w 1 m m m
 k w m v w
 — 1 1
 — 1 1 1 1
 i i v m w i i
 i i i i w i i w m v
 w v i i m k m m w
 m m v v m
 w m m Prod m m w fl
 m m v m k m w v w m m w
 m v m k w m w
 v m w m w

5 Conclusion

m w v w m

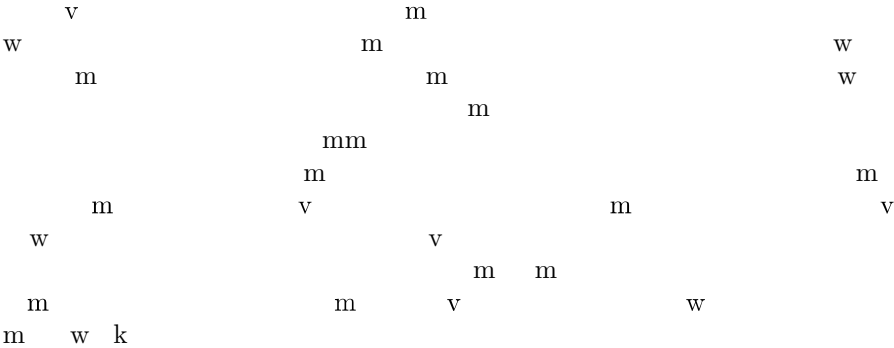
m v m mm

m w

m

1 1	1 9 99	1 1 91 1 9	1	1	99 9	
1		11		11	1 1 9 9	1 9 1 1 1
1	1	99 1	9 1	9 9 1	1 1	1 9
	9		9 11 1		1 9 9 9	9 1 9

Table 2. *LT*_L m v



References

1

Advances in Petri nets

LNCS

19

High-level Petri Nets, Theory and Application

99

1991

Information Processing Letters

199

EATCS-An Introduction to Petri Nets

19

fl

Proceedings of the 5th International Conference on Computer Aided Verification, Greece

9

Lecture Notes in Computer Science

9

199

Advances in Petri Nets

1

Lecture Notes in Computer Science

9 9

199

Bisimulation and the Reduction of Petri Nets

Philippe Schnoebelen¹ and Natalia Sidorova²

¹ Lab. Spécification & Vérification, ENS de Cachan & CNRS UMR 8643
61, av. Pdt. Wilson, 94235 Cachan Cedex, France

`phs@lsv.ens-cachan.fr`

² Eindhoven Univ. of Technology, Faculty of Electrical Engineering
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

`natalia@ics.ele.tue.nl`

Abstract. We investigate structural equivalences on places of P/T nets that allow reductions compatible with bisimilarity. This comes with a study of two kinds of reductions: fusion of equivalent places, and replacement of some places by other ones. When effectivity issues are considered, we are lead to a variant of place bisimulation that takes into account a set of “relevant” markings.

1 Introduction

is a formal notion of equivalence between places of P/T nets that combines two distinctive features: it agrees with a notion of , and it relates equivalent rather than equivalent markings.

While several early papers adapted Milner’s concept of bisimilarity to the transition systems generated by Petri nets [NT84, Pom86], Olderog was the first to propose a notion of bisimulation that comes from lifting a bisimulation between places [Old89, Old91]. The main advantage of this approach is that, as with many structural methods on Petri nets, it considers the elements (places and transitions) of the net rather than the reachability graph (which may be infinite). Using Olderog’s method, it was possible to prove that two nets had equivalent behaviours just by exhibiting a relation between their places and checking a finite number of local constraints.

The name “place bisimulation” comes from [ABS91] where Olderog’s proposal is improved and where it was proven that place bisimulation could be used to P/T nets (rather than just state two nets have equivalent behaviours).

are transformations of Petri nets that decrease the size. Their main use is as an optimization technique. For such applications the reduction must return a net behaviorally equivalent to the original one, but smaller in size. This smaller net can be used in place of the original net when it comes to implementing the net, or to analyzing it (e.g., for verification purposes).

Place bisimulations yield reductions that preserve the branching-time behaviour. These reductions are (one removes places and transitions,

and this is what may remove markings) and do not depend on the initial marking (or the set of reachable markings).

The fact that place bisimulations do not depend on the set of reachable markings has its pros and cons. On the positive side, this allows polynomial-time algorithms, and leads to reductions that are correct whatever the initial marking. On the negative side, some reductions that seem obviously correct are not given by place bisimulation simply because there exists an (irrelevant) alternative initial marking for which they are incorrect. This is the drawback we try to alleviate in this paper.

In this paper we investigate structural reductions of P/T nets that preserve bisimilarity and that take into account a notion of which markings are relevant. Our starting point is an investigation of what are the correct ways of defining an equivalence on places. This departs from earlier works where behavioural equivalence is the starting point, and reductions only a byproduct.

We first identify two different, equally natural, reductions based on a notion of equivalent places. They are “fusion” and “replacement”. There is a strong connection between them but correct fusions only coincide with correct replacements when we impose (rather strong) structural conditions on the set of relevant markings. These theoretical results and the accompanying examples are enlightening and they help understand why it is quite delicate to take into account a set of markings in these structural reduction problems.

Then, decidability issues impose further restrictions. We end up with a variant of place bisimulation that is parameterized by a set of markings, a partial answer to what was felt as the main inconvenience of place bisimulation. We say the answer is only partial because the set of reachable markings has to satisfy some strong closure restrictions.

We consider bisimilarity as our basic correctness notion because it has proven to be a fundamental semantic equivalence in the algebraic theory of concurrent systems [Mil89, Mil90, Gla90], because it is more local than more classical language-theoretical notions and then often behaves better algorithmically, because there exists a very successful verification technology for transition systems based on bisimilarity (e.g., [CPS93]), because it has not been much studied in the context of Petri nets.

The work in [ABS91] was continued in [AS92] (studying how true concurrency is preserved, and where an algorithm for place bisimulation is given) and [APS94] (investigating place bisimulations abstracting from silent moves). These works did not start from reduction issues and we find our new approach clearer and more natural (also, these works did not try to take into account a set of relevant markings).

Independently, Voorhoeve studied a concept very similar to place bisimulations [Voo96]. His starting point is also “bisimulation + structural equivalences”,

but he emphasizes equivalence of behaviours and does not mention applications to reductions of Petri nets.

For optimization purposes there also exists structural reductions that do not preserve bisimilarity (e.g., [NKML95, STMD96]). For more specific purposes, reductions have also been used as a computational device [Ber86]. E.g., telling whether a net is bounded can be done by reducing it to a normal form where boundedness is obvious. Here it is not necessary that the reductions preserve all of the behaviour of the net, only the relevant aspect (boundedness in our example) is enough.

We first recall basic definitions and notations about P/T nets and bisimilarity (2). Then we define reductions by fusion (3) and by replacement (4). It is then possible to define and study when a correct replacement gives rise to a correct fusion and vice versa (5). We investigate how correct reductions combine (6) and show there exists a largest correct reduction, unfortunately not computable in general (7). Place bisimulation is then introduced as a computable approximation (8) and exemplified (9).

2 Basic Definitions

\mathbb{N} denotes the set of natural numbers. Let $\Sigma = a, b, c, \dots$ be a finite alphabet of symbols.

2.1 Labeled Nets and Their Behaviour

A Petri net is a tuple $N = (P_N, T_N, F_N, l_N)$, where:

- P_N and T_N are two disjoint non-empty finite sets of places and transitions respectively;
- $F_N : (P_N \times T_N) \cup (T_N \times P_N) \rightarrow \mathbb{N}$ is a flow function (also called f);
- $l_N : T_N \rightarrow \Sigma^*$ labels each transition $t \in T_N$ with some action $l_N(t)$ from Σ^* .

We drop the N subscript whenever no ambiguity can arise and present nets with the usual boxes and circles graphical presentation.

Markings are configurations of a net. A marking M of N can be seen as a multiset over P , i.e. a mapping from P into \mathbb{N} . This is the viewpoint we adopt here. Then $M(p)$ is the number of appearances of place p in multiset M : this represents the number of tokens on p in marking M . The set of all possible markings is \mathbb{N}^P .

We write $|M|$ for the size of a marking M (its number of tokens) and, given two markings M, M' we let $\Delta(M, M')$ denote the distance between M and M' , i.e. the number of tokens one has to move to transform M into M' .

Given a transition $t \in T$, the $\text{dom}(t)$ and the $\text{codom}(t)$ of t are the multisets of places given by $\text{dom}(t) \stackrel{\text{def}}{=} F(p, t)$ and $\text{codom}(t) \stackrel{\text{def}}{=} F(t, p)$ for any $p \in P$.

A transition $t \in T$ is *enabled* in marking M iff $\forall p \in \text{dom}(t), M(p) \geq F_N(p, t)$. An enabled transition t may fire, thus performing action $l(t)$. This results in a new marking M' defined by $M' \stackrel{\text{def}}{=} M - \text{dom}(t) + \text{codom}(t)$ (that is, $M'(p) \stackrel{\text{def}}{=} M(p) - F(p, t) + F(t, p)$ for any $p \in P$).

We write $N : M \xrightarrow{t} M'$ to denote that $M \xrightarrow{t} M'$ is a step in net N . Derived notations are “ $M \xrightarrow{t} M'$ ” when N is implicit, “ $M \xrightarrow{a} M'$ ” when a is $l(t)$, “ $M \xrightarrow{a}$ ” when M' is not relevant, etc.

2.2 Bisimulation

Let $N_1 = (P_1, T_1, F_1, l_1)$ and $N_2 = (P_2, T_2, F_2, l_2)$ be two nets and $R \subseteq \mathbb{N}^{P_1} \times \mathbb{N}^{P_2}$ be a relation between their markings.

We say that R has the *transfer property* iff for all M_1, M_2 s.t. $M_1 R M_2$, and for all steps $N_1 : M_1 \xrightarrow{t_1} M'_1$, there exists a step $N_2 : M_2 \xrightarrow{t_2} M'_2$ s.t. $l(t_2) = l(t_1)$ and $M'_1 R M'_2$.

If R and R^{-1} have the transfer property (we sometimes say that R has the transfer property in both directions) then R is a *bisimulation* between N_1 and N_2 [Par81, Mil89]. When $M_1 R M_2$ for a bisimulation R we say that (N_1, M_1) and (N_2, M_2) are *bisimilar*, written $N_1, M_1 \sim N_2, M_2$. When N_1 and N_2 are the same net, we simply write $M_1 \sim M_2$.

It is well-known that, given any two nets, there exists a largest bisimulation between them, and we use $\sim_{[N_1, N_2]}$ to denote it. When $N_1 = N_2$, the largest bisimulation is an equivalence. Our proofs that relations are bisimulation often use basic properties (like $\sim_{[N_2, N_3]} \circ \sim_{[N_1, N_2]} = \sim_{[N_1, N_3]}$) and Milner’s “up to” technique [Mil89]: we say R has the transfer property *if* $M_1 R M_2$ and $M_1 \xrightarrow{t_1} M'_1$ imply that there exists a $M_2 \xrightarrow{t_2} M'_2$ with $M'_1 R M'_2$ for $R \stackrel{\text{def}}{=} \sim_{[N_2, N_2]} \circ R \circ \sim_{[N_1, N_1]}$ (and $l(t_2) = l(t_1)$). Then R has the transfer property.

3 Reduction by Fusion

is a natural way of reducing nets, based on the general idea of quotients of structures.

Consider a net N and assume $B \subseteq P \times P$ is an equivalence relation between places. Then N can be reduced by fusing B -equivalent places into a single place. This yields a new net denoted N/B . We write p/B for the equivalence class of p (and then for the fused place in N/B). The transitions are untouched but the arcs between places and transitions follow the fusion process: formally, if t (resp. t') is p_1, \dots, p_k in N , then in N/B , t (resp. t') is $p_1/B, \dots, p_k/B$. A marking M in N is fused into a marking $m \stackrel{\text{def}}{=} M/B$ in N/B . m and M have the same number of tokens. Fig. 1 gives an example where B relates p_2 and p_4 .

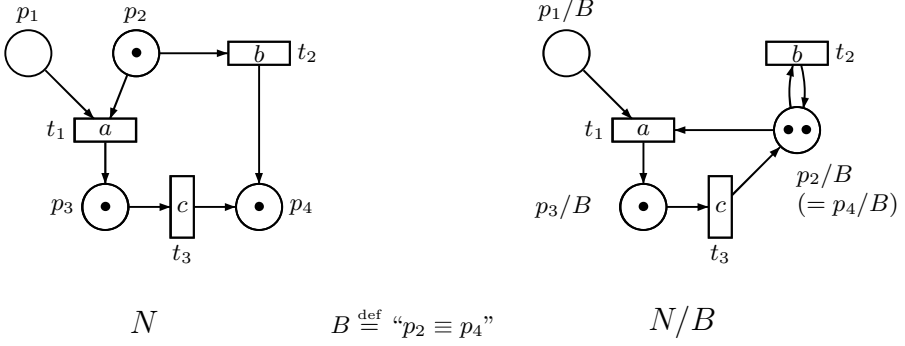


Fig. 1. Fusion of places as a reduction method

Fusing places modifies the behaviour, but only in one direction: it adds new behaviours. This is captured as

Fact 3.1. $M \stackrel{t}{\rightarrow} M' \quad N \quad M/B \stackrel{t}{\rightarrow} M'/B \quad N/B$

This implies that $_{/B}$, defined as $(M, M/B) \in \mathbb{N}^P$, has the transfer property from N to N/B .

A fundamental property is the following weak converse:

Proposition 3.2. $m \stackrel{t}{\rightarrow} m' \quad N/B \quad M \stackrel{t}{\rightarrow} M' \quad M/B = m' \quad m = M/B \quad m' = M'/B \quad M \stackrel{t}{\rightarrow} M'$

Assume $m \stackrel{t}{\rightarrow} m'$. Write M_1 for t in N and pick any M_2 s.t. M_2/B is $m - (M_1/B)$. Then, in N , $M \stackrel{t}{\rightarrow} M'$ for $M = M_1 + M_2$ and $M' = t + M_2$. Clearly, $M/B = m$.

An equivalence B between places can be lifted to an equivalence, written \overline{B} , between markings. Formally, if $p_1 B p_1, \dots, p_n B p_n$ then $p_1, \dots, p_n \overline{B} p_1, \dots, p_n$. Note that $M \overline{B} M'$ iff $M/B = M'/B$, i.e.

$$\overline{B} = (_{/B})^{-1} \quad (_{/B}). \quad (1)$$

Now, a direct reading of Prop. 3.2 gives the following

Fact 3.3. $\overline{B} \quad \overline{B} \quad [N, N]$
 $(_{/B})^{-1}$

4 Reduction by Replacement

is a second natural way of reducing nets, this time based on the general idea of homomorphic images of structures.

Consider a net N and assume $h : P \rightarrow P$ is a projection, i.e. for all $p \in P$ we have $h(h(p)) = h(p)$. In N a place p can be replaced by $h(p)$ whenever $h(p) = p$. This yields a new net, denoted $h(N)$.

More precisely, $h(N)$ is obtained by redirecting all output edges of transitions by their projections: if $t = M = \langle p_1, \dots, p_k \rangle$ in N , it becomes $h(M) \stackrel{\text{def}}{=} \langle h(p_1), \dots, h(p_k) \rangle$ in $h(N)$. All places not in $h(P)$ are removed, all transitions that have some input place removed are removed as well. A marking M in N yields a marking $m = h(M)$ in $h(N)$. m and M have the same number of tokens. Fig. 2 gives an example where p_3 , and then t_3 , are removed.

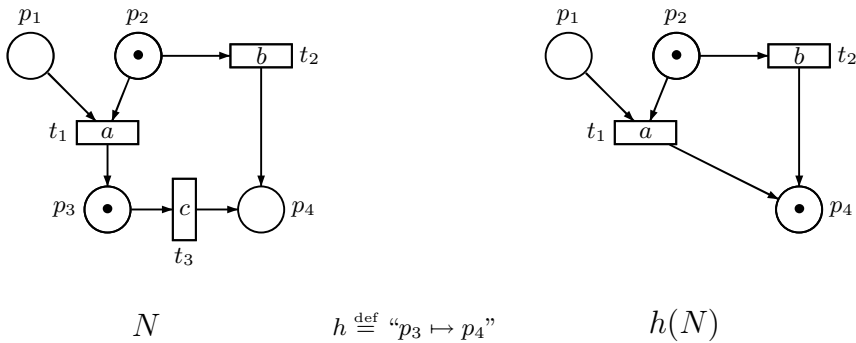


Fig. 2. Replacement of places as a reduction method

Replacing places modify the behaviour in the following way:

Fact 4.1. $m \xrightarrow{t} m' \text{ in } h(N) \iff m \xrightarrow{t} M \text{ in } N \text{ and } M \xrightarrow{h} h(M) = m'$

There is a weak converse:

Fact 4.2. $M \xrightarrow{t} M' \text{ in } N \text{ and } h(M) = M' \implies M \xrightarrow{t} h(M) \text{ in } h(N)$

5 Correct Reductions

We are only interested in $\text{correct reductions}$. Informally, a correct reduction is a reduction such that the reduced net N is a correct variant of the original net N .

In this paper, we investigate formal notions of correctness

1. grounded into bisimilarity, and
2. only applying to a given set of markings (called correct variants).

5.1 Relevant Markings

In all the following, we assume \mathbb{N}^P is a set of relevant markings. Typical examples are “the set of all markings reachable from some initial M_0 ”, or “all markings with three tokens”.

Compared to earlier works on place bisimulation, we remove the basic (implicit) assumption that all and every markings of the nets under study are meaningful. It was a very strong assumption, resulting into a very conservative view of when two places are bisimilar.

This is why our new assumption is that we only aim at correction relative to a given set of relevant markings. In a sense, this work can be seen as extending the earlier place bisimulation theory to a framework with relevant markings. But, in addition, the paper also brings a new viewpoint: here we focus on reductions while the earlier works focused on semantical equivalences. We think the new viewpoint is clearer and more natural.

5.2 Correctness for N with \mathcal{A}

Assume N comes with a set of relevant markings. A reduction is correct if all relevant markings are reduced into bisimilar markings in the reduced net. The other (= irrelevant) markings are not considered for the correctness issues.

With this in mind, the following definitions are natural:

Definition 5.1.
$$\begin{array}{ccc} B & P & P \\ N, M & N/B, M/B & M \end{array} \quad \begin{array}{l} \text{correct fusion} \\ \end{array} \quad N$$

Definition 5.2.
$$\begin{array}{ccc} h : P & P & \\ N, M & h(N), h(M) & M \end{array} \quad \begin{array}{l} \text{correct projection} \\ \end{array} \quad N$$

We are interested into the connections between these two definitions. Indeed, any projection $h : P_N \rightarrow P_N$ naturally induces an equivalence B_h given by $pB_hp \stackrel{\text{def}}{=} h(p) = h(p)$. Can we say that B_h is a correct fusion iff h is a correct projection? It turns out this is not the case in general.

Fact 5.3.
$$\begin{array}{ccc} h & & N, \\ B_h & & \end{array}$$

Figure 3 gives an example where $h \stackrel{\text{def}}{=} p_3 \rightarrow p_1, p_4$ is correct (for the set of reachable markings) but B_h , “ $p_1 \rightarrow p_3 \rightarrow p_4$ ”, is not.

Another example is provided in Figure 4 where $h \stackrel{\text{def}}{=} p_3 \rightarrow p_2, p_4$ and $p_1 \rightarrow p_1$.

Fact 5.6.
$$\begin{array}{ccc} h & B = B_h & B \end{array}$$

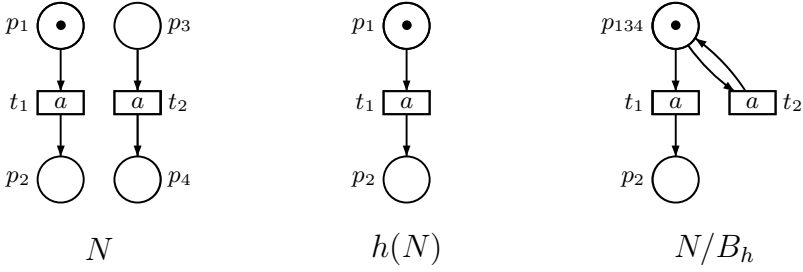


Fig. 3. $h = p_3 \ p_1, p_4$ p_1 is correct and B_h , “ $p_1 \ p_3 \ p_4$ ”, is not

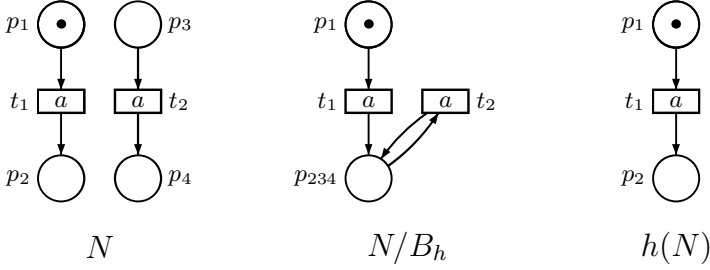


Fig. 4. $h = p_3 \ p_2, p_4$ p_2 is correct and B_h , “ $p_2 \ p_3 \ p_4$ ”, is not

Figure 5 gives an example where B , “ $p_1 \ p_2$ ”, is a correct fusion (for the set of reachable markings) but where no h is correct (the figure illustrates $h \stackrel{\text{def}}{=} p_1 \ p_2$).

Another example is provided in Figure 6 with B given by “ $p_2 \ p_3$ ” and $h \stackrel{\text{def}}{=} p_2 \ p_3$. Here $\stackrel{\text{def}}{=} p_1$.

It turns out that all these difficulties come from the sets we picked in these examples. They disappear when we impose restrictions on possible choices for

Given N and a set of markings, we say that is, if M and $M \dashv M$ entail M . The sets in examples 5.4 and 5.7 are succ-closed by construction.

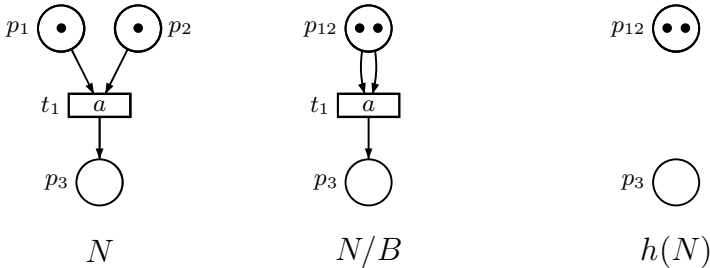


Fig. 5. B , “ $p_1 \ p_2$ ”, is correct and $h = p_1 \ p_2$ is not

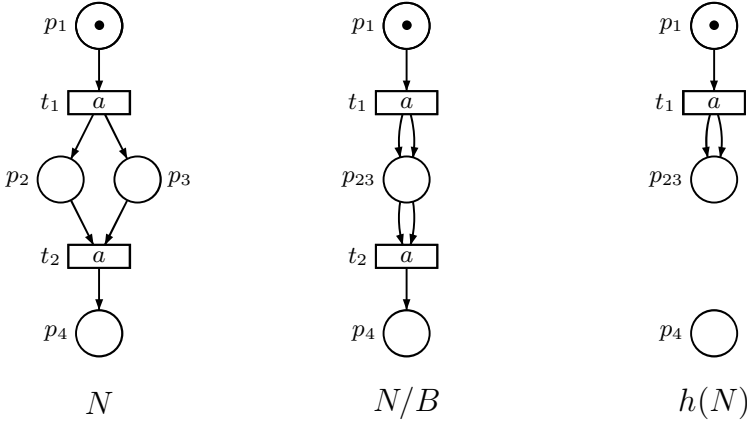


Fig. 6. B , “ $p_2 \ p_3$ ”, is correct and $h = p_2 \ p_3$ is not

Secondly, given an equivalence B , we say that Σ is B -closed if $M \in \Sigma$ and $M\overline{B}M$ entail $M \in \Sigma$. The sets Σ in examples 5.5 and 5.8 are B -closed.

We can now state the following crucial property, explaining how when Σ is succ-closed and B -closed, checking whether B is correct can be done by inspecting N only. This lemma will be of great help because, in general, it is difficult to tell whether a given B or h are correct: indeed, for this we have to compare behaviours in N and in N/B , or $h(N)$.

Lemma 5.9.

$$\overline{B} \cap (\Sigma) \subseteq \Sigma$$

N

() This is obvious when $\Sigma = \mathbb{N}^{P_N}$, using (1) and the assumption that Σ/B is a bisimulation. In the general case, B -closure of Σ gives that $\overline{B} \cap (\Sigma) = R^{-1} R$ where $R \stackrel{\text{def}}{=} (M, M/B) \in \Sigma$ is included into $[\Sigma, N/B]$ (by assumption).

() Now assume $\overline{B} \cap (\Sigma)$ is included in Σ , and define $R \stackrel{\text{def}}{=} (M, M/B) \in \Sigma$. Fact 3.1 and succ-closure of Σ implies that R has the transfer property.

For the other direction, the reasoning underlying Fact 3.3 applies even when Σ is taken into account. As a consequence, R is included into $[\Sigma, N/B]$, so that B is a correct fusion.

Theorem 5.10.

$$h \text{ is a correct projection} \iff N, B_h \text{ is a correct fusion} \iff h(N), h(M_1) \in N, M_2$$

() Assume h is a correct projection and consider $M_1 \in N$. Let $M_2 \in N$ s.t. $h(M_1) = h(M_2)$. Correctness of h implies $N, M_1 \in h(N), h(M_1) \in N, M_2$ (since $M_2 \in N$), hence $M_1 \in M_2$. Thus $\overline{B}_h \cap (\Sigma) \subseteq \Sigma$ and Lemma 5.9 concludes.

() Now assume that B_h is a correct fusion. Lemma 5.9 and the fact that $M\overline{B_h}h(M)$ give that $N, M \rightarrow h(N), h(M)$.

Our earlier examples show that succ-closure or B -closure alone is not sufficient for Theorem 5.10.

When \rightarrow is the set of all markings, closure is guaranteed and fusions or replacements coincide. In the following, whenever \rightarrow is succ-closed and B -closed, we speak of correct equivalences without more precision on whether we mean fusions or replacements.

6 Combining Correct Reductions

Correct reductions can be composed, but the natural ways for composing projections and for composing fusions are not the same.

Lemma 6.1.
$$h_1 \rightarrow N, \quad h_2 \rightarrow h_1(N), h_1() \quad h_1() \stackrel{\text{def}}{=} h_1(M) \rightarrow M \quad) \quad h_2 \rightarrow h_1$$

Direct from Def. 5.2 and transitivity of bisimilarity.

Furthermore, if \rightarrow is succ-closed and B_{h_1} -closed, and $h_1()$ is B_{h_2} -closed, then \rightarrow is succ-closed and $B_{h_2 \circ h_1}$ -closed.

Now let B_1 and B_2 be two equivalences and assume \rightarrow is B_1 -closed, B_2 -closed and succ-closed. Then

Lemma 6.2.
$$B_1 \rightarrow B_2 \rightarrow N, \quad B \stackrel{\text{def}}{=} (B_1 \rightarrow B_2) \rightarrow N,$$

Once we notice B -closure of \rightarrow and $\overline{B} = (\overline{B_1} \rightarrow \overline{B_2})$, Lemma 5.9 does all the work.

A related question is whether, assuming B is a correct fusion, all more conservative reductions are correct too? Here, the intuition underlying the question is that an equivalence relation can only be incorrect by making too many identifications, not by making too few.

Let B be a correct fusion for some N , (assuming \rightarrow is B -closed and succ-closed), then

Lemma 6.3.
$$B \rightarrow B \rightarrow B \rightarrow B$$

Use Lemma 5.9 and $\overline{B} \rightarrow \overline{B}$.

This question can also be investigated in terms of projections: let h be a correct projection and $h = h_2 \circ h_1$. Is h_1 correct?

Lemma 6.4.
$$h = h_2 \circ h_1 \rightarrow N, \quad B_h \rightarrow h_1 \rightarrow N,$$

First, note that $B_{h_1} \rightarrow B_h$ and that \rightarrow is B_{h_1} -closed. Now, if h is a correct projection then (Theorem 5.10) B_h is a correct fusion, hence (Lemma 6.3) $B_{h_1} \rightarrow B_h$, hence (Theorem 5.10) h_1 is a correct projection.

Definition 8.4. $\text{minimal covers } t \text{ } C(t)$
 $M \quad t \quad M$

Here “ M minimal” means that M and $t \quad M \quad M$ imply $M = M$.

Note that Dickson’s lemma implies the finiteness of $C(t)$ for any t . $C(t)$ can be empty if there is no way to enlarge t into some element of \quad .

Definition 8.5. $B \quad P \quad P$ weak transfer property
 $B \quad t \quad T \quad p \quad t \quad q \quad P \quad pBq$
 $M_1 \quad C(t) \quad u \quad T \quad l(u) = l(t) \quad M_2 \quad M_1 - p + q$
 $M_2 \quad u \quad M_2 \quad M_1 \overline{B} M_2 \quad M_1 \quad t \quad M_1$

Hence the weak transfer property is the transfer property restricted to pairs M_1, M_2 and steps $M_1 \quad t \quad M_1$ where M_1 is in the minimal cover $C(t)$ and where M_2 differs from M_1 by only one token. Since this is a weaker property, we have

Fact 8.6. $B(N, \quad)$

Lemma 8.7. $B \quad B \quad B$

We show that $\overline{B} \cap (\quad)$ has the transfer property, i.e. “for all $M_1 \overline{B} M_2$, for all $M_1 \quad t_1 \quad M_1$, there is a $M_2 \quad t_2 \quad M_2$ s.t. ...”, by induction over the pair $(M_1, \Delta(M_1, M_2))$ ordered lexicographically.

So let us assume $M_1 \quad t_1 \quad M_1$ and $M_1 \overline{B} M_2$. Write n for $\Delta(M_1, M_2)$. We distinguish four cases:

1. $n = 0$: then $M_2 = M_1$ and we are done.
2. $n = 1$ and $M_1 \quad C(t_1)$: then we are exactly in the situation where the weak transfer property applies.
3. $n > 1$: then there is a $M_3 \quad$ s.t. $M_1 \overline{B} M_3 \overline{B} M_2$ and $\Delta(M_i, M_3) < n$ for $i = 1, 2$. By ind. hyp., we can transfer $M_1 \quad t_1 \quad M_1$ to some $M_3 \quad t_3 \quad M_3$ that we can then transfer to some $M_2 \quad t_2 \quad M_2$. We have $M_1 \overline{B} M_3 \overline{B} M_2$, hence $M_1 \overline{B} M_2$.
4. $M_1 \quad C(t_1)$: then there is a $M_{1,1} \quad C(t_1)$ s.t. $M_{1,1} \quad M_1$. We decompose M_1 as $M_{1,1} + M_{1,2}$ and M_2 as $M_{2,1} + M_{2,2}$ s.t. $M_{1,i} \overline{B} M_{2,i}$ for $i = 1, 2$. $M_{1,1} \quad t_1 \quad M_{1,1}$ and the ind. hyp. transfers this into a $M_{2,1} \quad t_2 \quad M_{2,1}$ with $M_{1,1} \overline{B} M_{2,1}$. We complete into $M_2 \quad t_2 \quad M_2 \stackrel{\text{def}}{=} M_{2,1} + M_{2,2}$ and have $M_1 \overline{B} M_2$.

We have the immediate corollary:

Theorem 8.8. $\quad \quad \quad B$
 $B(N, \quad)$

Finiteness of $C(t)$ implies that checking whether a given B has the weak transfer property only involves a finite number of checks. Therefore the following abstract algorithm computes $B(N, \quad)$:

Algorithm 8.9. a labeled Petri net N , and a succ-closed set of relevant markings.

$B(N, \text{ })$.

Let $B = E(N, \text{ })$, “the largest place equivalence under which is closed”.

Check if B has the weak transfer property:

- If it has, then B is $B(N, \text{ })$.
- Otherwise, there is a $t \in T$, a $M \in C(t)$, a $p \in M$, a $q \in P$ with pBq s.t. $M \xrightarrow{t} M$ can not be imitated from $M - p + q$. Then remove the pairs (p, q) and (q, p) from B and return to step 2.

This algorithm is obviously correct and stops after a finite number of steps. Its complexity depends on the size of $C(t)$, and the cost of computing $C(t)$ and $E(N, \text{ })$.

Note that it is possible to run the algorithm starting with any equivalence E included in $E(N, \text{ })$ (i.e. any E such that is E -closed), and obtain the largest place bisimulation included in E , a safe approximation of $B(N, \text{ })$.

9 Place Bisimulation in Practice

When we want to use Algorithm 8.9 in practice, we face two problems: we need some effective way of computing $C(t)$ for $t \in T$, and we need to compute $E(N, \text{ })$ that will be the first upper approximation of $B(N, \text{ })$.

Whether this can be done depends on how is given. In the simple case where is finite, then $C(t)$ and $E(N, \text{ })$ are computable in a direct way. More interestingly, when is given by a Presburger formula (i.e., is a semilinear set) then $C(t)$ and $E(N, \text{ })$ can then be defined by Presburger formulas, and can be effectively computed: indeed assume $P = \{p_1, \dots, p_k\}$ and is given by the Presburger formula $\psi(n_1, \dots, n_k)$. Then $(p_i, p_j) \in E(N, \text{ })$ iff $\psi(n_1, \dots, n_k)$ $n_i > 0$ entails $\psi(n_1, \dots, n_i - 1, \dots, n_j + 1, \dots, n_k)$.

As far as we could judge by dealing with a few examples, the equivalences we compute with Algorithm 8.9 are not much of an improvement over the classical notion of place bisimulation where all markings are considered as relevant.

The reason seems to be that, when using Algorithm 8.9, we first have to give a succ-closed set . If this is a large set, then it contains many irrelevant markings and this defeats the purpose of our study. If is as small as possible (e.g., it contains only the reachable markings) then this may impact the equivalence $E(N, \text{ })$ computed in and make it quite drastically limited even before it is refined into a place bisimulation.

In the end, we can only argue the usefulness of our definition with ad-hoc examples, so that it seems like it is mostly a theoretical contribution to place bisimulation. Our hope is that this contribution can be combined smoothly with

the ideas from [APS94] where silent moves are abstracted from, and that this combination does work well in practice.

10 Conclusion

This paper proposed a new point of view on Petri net reductions based on the bisimilarity notion, and that do not aim at correctness for all markings.

We started with a net N and looked at reductions that lead to bisimilar configurations for any of the markings in some set \mathcal{M} . It turns out restrictions must be imposed on \mathcal{M} in order to get a smooth theory.

Since the largest correct reduction is in general not computable, place bisimulations is an elegant close approximation. We gave an abstract algorithm computing the largest place bisimulation when \mathcal{M} is given in a sufficiently effective way.

These results shed a new light on earlier works on place bisimulation. Having reductions in mind from the start help understand why place bisimulation cannot cope easily with arbitrary set of markings. Indeed, we end up with quite strong restrictions on the set \mathcal{M} which, ideally, we would like to consist of the reachable markings only.

Acknowledgments

This paper greatly benefited from the advices of anonymous referees who suggested several simplifications of our earlier proofs, and who pinpointed some of the reasons why the method is hard to use in practice.

References

- [ABS91] C. Autant, Z. Belmesk, and Ph. Schnoebelen. Strong bisimilarity on nets revisited. In *Proc. Parallel Architectures and Languages Europe (PARLE'91)*, Eindhoven, NL, June 1991, vol. II: *Parallel Languages*, volume 506 of *Lecture Notes in Computer Science*, pages 295–312. Springer, 1991.
- [APS94] C. Autant, W. Pfister, and Ph. Schnoebelen. Place bisimulations for the reduction of labeled Petri nets with silent moves. In *Proc. 6th Int. Conf. on Computing and Information (ICCI'94)*, Trent University, Canada, May 1994, pages 230–246, 1994. Available at <http://www.lsv.ens-cachan.fr/Publis/PAPERS/APS-icci94.ps>.
- [AS92] C. Autant and Ph. Schnoebelen. Place bisimulations in Petri nets. In *Proc. 13th Int. Conf. Application and Theory of Petri Nets (ICATPN'92)*, Sheffield, UK, June 1992, volume 616 of *Lecture Notes in Computer Science*, pages 45–61. Springer, 1992.
- [Ber86] G. Berthelot. Checking properties of nets using transformation. In *Advances in Petri Nets 1985*, volume 222 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 1986.

- [CPS93] R. Cleaveland, J. Parrow, and B. Steffen. The Concurrency Workbench: A semantics-based tool for the verification of concurrent systems. *ACM Transactions on Programming Languages and Systems*, 15(1):36–72, 1993.
- [Gla90] R. J. van Glabbeek. The linear time – branching time spectrum. In *Proc. Theories of Concurrency (CONCUR'90)*, Amsterdam, NL, Aug. 1990, volume 458 of *Lecture Notes in Computer Science*, pages 278–297. Springer, 1990.
- [Jan95] P. Jančar. Undecidability of bisimilarity for Petri nets and some related problems. *Theoretical Computer Science*, 148(2):281–301, 1995.
- [Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall Int., 1989.
- [Mil90] R. Milner. Operational and algebraic semantics of concurrent processes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, vol. B, chapter 19, pages 1201–1242. Elsevier Science, 1990.
- [NKML95] M. Nakagawa, S. Kumagai, T. Miyamoto, and D.-I. Lee. Equivalent net reduction for firing sequence preservation. *IEICE Trans. on Fundamentals E.*, 78-A(11):1447–1457, 1995.
- [NT84] M. Nielsen and P. S. Thiagarajan. Degrees of non-determinism and concurrency: A Petri net view. In *Proc. 4th Conf. Found. of Software Technology and Theor. Comp. Sci. (FST&TCS'84)*, Bangalore, India, Dec. 1984, volume 181 of *Lecture Notes in Computer Science*, pages 89–117. Springer, 1984.
- [Old89] E.-R. Olderog. Strong bisimilarity on nets: a new concept for comparing net semantics. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Proc. REX School/Workshop, Noordwijkerhout, NL, May-June 1988*, volume 354 of *Lecture Notes in Computer Science*, pages 549–573. Springer, 1989.
- [Old91] E.-R. Olderog. *Nets, Terms and Formulas*, volume 23 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge Univ. Press, 1991.
- [Par81] D. Park. Concurrency and automata on infinite sequences. In *Proc. 5th GI Conf. on Theor. Comp. Sci., Karlsruhe, FRG, Mar. 1981*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 1981.
- [Pom86] L. Pomello. Some equivalence notions for concurrent systems. An overview. In *Advances in Petri Nets 1985*, volume 222 of *Lecture Notes in Computer Science*, pages 381–400. Springer, 1986.
- [Qui95] W. Quivrin-Pfister. *Des bisimulations de places pour la réduction des réseaux de Petri*. Thèse de Doctorat, I.N.P. de Grenoble, France, November 1995.
- [STMD96] S. M. Shatz, S. Tu, T. Murata, and S. Duri. An application of Petri net reduction for Ada tasking deadlock analysis. *IEEE Trans. Parallel and Distributed Systems*, 7(12):1309–1324, 1996.
- [Voo96] M. Voorhoeve. Structural Petri net equivalence. Tech. report, Eindhoven Technical University, Dept. Computer Science, 1996. Available at <ftp://ftp.win.tue.nl/pub/techreports/wsinmar/>.

Efficiency of Asynchronous Systems That Communicate Asynchronously

t g *

" m "

m

vogler@informatik.uni-augsburg.de

Abstract. m m

m m m

fl

m m m m

m m

m m m

m

m m

1 Introduction

th t ti g i i y y t

 y th i th y i i i t th y

t g t th t i t ith th t

thi *communication* i *synchronous* i i t ti

h i i hi h ti i th i i ti

ti thi *behaviour* i *asynchronous* hi i h h

th i g g i t th t

i y y h y t i g t i t y t t

y y h i ti t i i g t y gi g

t iti

th t t i t it i ti y t t i

th iti y gi g hi h t *asynchronous commu-*

nication hi iti h y tt ti t g

h g i g

th ty t i t t i y h i ti i th

g th thi

i h t t th t ti g h t ti h i i t

ti g ith y h i ti g th i

t t t thi i i y t t

i g i t t y h i ti i gi g

hi iti t t y th x h g g

★

t th fl t i i t i g th ti

th h t gi xi i th t ti

t it i i t th g it y t g h

g ti th th h i y i ti it i i t

ig t i t N th i t t g i t t t

h tt y t i ti N and th i t th t

tt ith N i p i g i t N th t N i t

t h t p th i ti t t h

p g g ti y i th t hi i i th

thi t y t g fl xi ty i g th i t

i gg t i ig t i t p t N i t t t

or both h ty t ti t th i t

hi hi t h t p p th h th i t y

i t N t t N th t iti y

N y g t h i i th i t ti th

ti h i ti N t p i

i t th i t hi h i h it ti

hi y th i h ith xi th

h h i t i ti i i i

t t th t t i t h ti y

th h h i t i ith i t t t

t x i th xt ti i t t th t it i t

g g t t ith

th i t ti g h y t i i t ti i it

i i t i ti y t th

i ti t i t t th i y i t ti

i ti i i t y i g t t ω h t

h ithi gi ti h i t t i t h i

tt ti g y iti t y h y t

ithi t i t i ti t t i t t

t t thi i i t t h t y it i y

thi i t t ti th t i i y th t thi gi

g ti h y t ti i t h i th y

t ti g

y t th t h ti i ithi

gi ti i i ithi thi ti h ti

i i gi g th i y g y i

ti it i y t th t ith i

t i t ti thi ti h i

th y y h y t hi i g h h

t iti h t ithi ti i h t iti th t h t

i i t y i ithi ti thi t t g ity t

i g thi g i t th ti h x i ity i

i ti

t iti ith ty i g i
 i t i ti ti gi ti i g i g i t
 ti i y t ti g t N_1 t th t N_2 i it
 ti t t th t N_2 ti i i it xhi it t t th ti
 ity ith t t th i y N_2 i h th t i t
 i ti hi h i t i t th t t th i
 g th iti i g h g t
 i g hi i g i gi i ti h i t
 h t iz t th ith t i g t t t i t t h thi t
 t t tti g h th t iti h i
 t i t i t i th th h t iz ti
 i ti t x t th i i
 i g g i g th t ti t i g i ti th
 th ih ti i i i

2 Basic Notions of Petri Nets with Time Bounds and Interface

thi ti i t t i t t hi h xt ith
 ti th t ig t ht iti ti ith i t
 i t t t x i i th i t ti th
 i i g th iti h t
 h *Petri net with time bounds and interface* N $S, T, W, \beta, I, O, M_N$
 t net h t i t it i i t t S places T
transitions th ti W S T T S i i g th arcs th
time bound β T , th input places I S th output places O S
 th initial marking M_N S N_0 i t ith N it t interface
places P I O t th t i t p ight i I O t th
 ti th t M_N i P h i t t N N_1 t
 th th t i i it y thi i t it t g S I P
 S_1 I_1 P_1 t
 h x S T th preset x i x y W y, x th postset
 x i x y W x, y h ti xt t t g
 X i th i x ith x X y y th x y loop
 marking i ti S N_0 ti g t h t i ti
 ti hi h th t th t t y h
 h g i g t t t iti th
 t i

Assumption: t i i thi t i t i
 s s h s S
 t iti x i i i x, y
 ith W x, y s i ith t iti t ith
 β t h t y th t iti i t s t
 t i t th x t i t i th x ti g t T t

h t t β t th gi th x i it ight i i
th t i 1-transition th i t i th
 β t t i 0-transition i t i i i t y iti g it
ty xt t it IO-place i th tt
x ig hi h i i
th i i g hi h ig ti i g i t
t iti t i enabled i g M t y M t i t M
M t M M t - t th t thi y M t M y th t
t occur fire M yi i g th i g M
hi iti i g xt t
w t iti i enabled i g M t
y M w yi th i g M h occurring t y
M w M i w λ M M w w t M w M M t M
i g M t iti t w i th i iti i g
th it i firing sequence th t th i t y FS N
i g M i reachable i M_N w M w T h t i
safe i M s s h i g M
i t th iti t t iso-
morphic i t i th th y i ti y i g th
t iti th i S P hi i g th t t ti
i iti i g g i hi t y
th i tity th i t i i t t h h t
t N_1 N_2 th t th y i i t t i th i i t
i S_1 T_1 S_2 T_2 P_1 P_2
i i t N_1 N_2 ith th parallel composition th th y
i i t y h y th i i t
t t th t i g th i t
 N_1 N_2 t t th t N_1 N_2 ti y th
i i t iti th i it i hi i g i
t ti y iti y th ty i g th i t i t t t
t ti th ti th t g N_1
y y t t t i t p N_2 i p i i t
 N_2 h N_1 N_2 i i y i i, j , p P_i P_j
t T_j h th t W_j t, p i i p I_i th t W_j p, t i i
p O_i
th iti ti N N_1 N_2 i t i y t i
i x t th i t ti t th i th t
ti i th y i i t t th i t th
i iti i g i h th y th p th thi i
i t h th i g iti y I I_1 P_2 I_2
 P_1 I_1 I_2 O O_1 P_2 O_2 P_1 O_1 O_2
t x i th t t t th i t h i t N i
xt g h th i th i t N_1 N_2 th
ty i g i t p i i g y t i y N_1 N_2

p i g i I₁ t t i O₁ th N₁ th t it i t
t p th ti t y N th i U
N th t t t p i i t y N₁ i t it ti
p i g y i t N₁ y t t N₂ th th i
t p i N₁ p i N₂ thi x t y t th t i t
h t iti y gi g th t i i th i t ti
t th t h p i t i th i t N hi h i i i
U ti y th ti N₁ N₂ p i t t i t
g N₂ t N₁ y i ti h

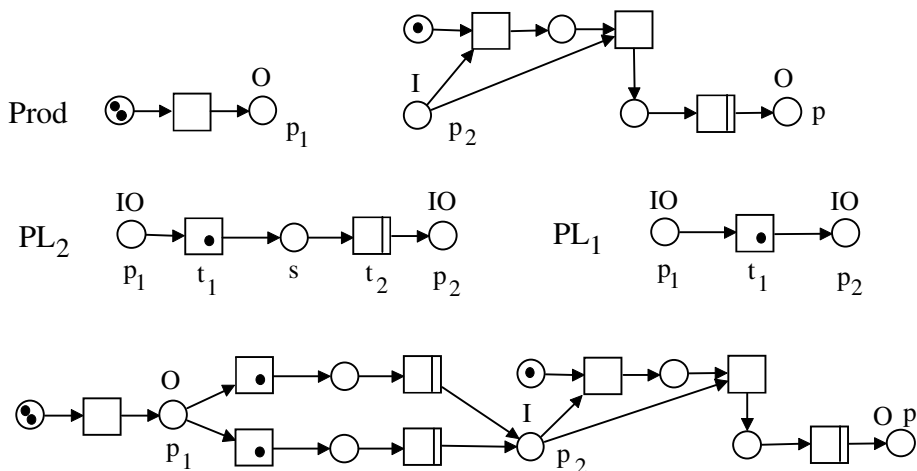


Figure 1

x t th i ti i ig Prod
i ty i it i i y i ith t iti tt it t
t t p₁ th iti i i g t t tt t it i t
p₂ t thi it it t iti i th t t t
ti th t t t p i t ig ti ti i i y th
i ω i i t t t ig i ti ti th
h ti i PL2 x t p₁ t
t t p₂ ti h t t g t iti th t h t
t ti h t ti i t ith th t th ti th t t g
h t t th th th th gh
th t g i t i g ti t g th iti
PL1 i i i PL2 i g y t g t i y x t
it t t
th i t ti iti y ti th t i t ty i g
t i t i t h t gi g t i th
it t h t x t t th th fl xi
i t ty i g ti i ti t Prod ith t ti i i t
th t Prod PL2 PL2 h i ig th t th

t i PL2 t ti y i i t x t th i i t
 t iti y x t PL2 PL2 t t th PL2 t th t th
 i y t t p₁ y PL2 i th th t
 thi i p₁ y i t PL2 t h i y Prod PL2
 PL2 t h it i y t h p₁
 i i y p₂
 y i y i t h i t th
 iti i y i th i h t
 gi g ti i th i t ti h t i t h t
 th h y i t h i t either i t
 ith y tg i g or t t ith y i g i g i
 g i t p t it tg i g p y g ith
 th t tg i g i it i t i ith t t
 ti i i g t th ti th t i th
 t ti t *I- or O-hiding* i t pi t N t i N/^Ip
 N/^Op hi hi t i N y i g p I O *hiding* pi
 y N/p N/^Ip/^Op *Renaming* i t pi t N t p P t
 i N p p hi hi t i N y i g p y p hi h i h it p
 i g ty i g i p S P it th t i hi y
 y h
 th i ti i PL2 PL2 i t th i g
 h it it PL2 PL2 /^Op₁/^Ip₂ th t
 t i PL2 PL2 t i h hi h y t g
 PL2 PL2 p₁ p₁ t ti i t th i p₂

3 Timed Behaviour of Asynchronous Systems and Testing

i i th y h h i y t t i g i t
 t th ti t hi h thi g h h t y h
 y t y i t th t th y g t t h
 ti ithi it gi ti thi ti th
 ti th t t y it i y i h iti
 ti h th h i i t y y h
 t t t t th t th ti i
 t iti t t t g ity th h t i g thi g i
 t th ti hi i t
 t iti ith it y ti i
 h ti t iti y t i g i t
 , tt h t th t iti i t g y
 tt h it y ti i t t th t t iti gg t
 i i y t ti g ith y h i ti i t i
 t ith h i t i ih it i h th t th t
 th ti t ti g i th tt h th ith

ti i t ti h i thi t
 t i t tti g x t th t g t th
 th ti i g i y i t ti hi h i i t h
 i h t g ti h i it i i t
 t i i t th i th it
 t iti i it y i t iti y h i g
 ti y y i y i g it i i t t t iti
 i it h t i i t y i ti i it h
 h i t iti g t it y it it
 ti it σ h *time step* t t h g t iti h
 iti y t th i g h y t h
 g t t th t iti h t i i t y i h i
 it ti ith t hi i t iti i iz
 h th t i iti y t i thi i t t i
 t iti t th t th i t
 i iti i g h t

Definition 1. *timed marking* TM t i i M, M^{old} i ti g
 t i g N ith M M^{old} g i i t i g g ti
 i g TM_1 i i it y i t M_1 M^{old}_1 t h i iti TM i TM_N
 M_N, M_N
 it M, M^{old} ε M, M^{old} i th i g i

$$\begin{aligned}
 \varepsilon & t \quad T, M t M, M^{old} \quad M^{old} \ominus t \quad h \quad n \ominus m \quad x \quad n - m, \\
 \varepsilon & \sigma, t \quad T \quad M t \quad \beta t \quad M^{old} t, M \quad M^{old} \quad M
 \end{aligned}$$

izi g thi ti i g t t g th ith
 ti h ti i g th t TFS N w $TM_N w$
timed firing sequences N ti i g w ζ w i th *duration*
 i th σ i w h h i i t t σ i *round*

t thi i g th t y t iti th t i
 it ti y t iti ti g t
 i g t t t iti y t t i g $PL2$ i
 ig ith t iti t p_1 hi h i t i t i
 thi i ti i t_1 i y t t y t
 t t t_1 i g t thi t ti i g ith t
 h p_1 s th t_1 h y th t p_1 i ith
 t_2 t_1 i y t h i i ty σ
 th t t t_1 t_2 th xt σ th
 i i ty i i t y t_1 t_2 t th t_1 i g t t
 t y y yi g g t FS N TFS N iti y th
 σ y h g M^{old} hi th i g t iti y
 M ti g th ti t i TFS N
 g t x t y FS N hi h th t it th ti ti

ith th xity y h y t h i y i h
th y h h i y ti i gi ti i th g y
th i g t i t h i ti y h
y t i i thi g itt i t iti i gi h i it y t
t ti i g h it ight t h
i i $M s > M^{old} s$ i i ti g th h t s
t s TM th th t thi i g $M^{old} s$
h g h gi h th M^{old} t
g th y th t y t thi
t t i t y y y h t ti
i g t gi iti ti i g h
ith i

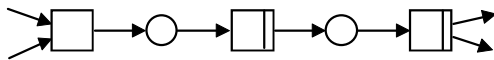


Figure 2

it g y t iti ith it y
th i duration g ig h t h th tt iti t
t t i i t y th y i hi th tt iti
t t t t tt ti t th th t t
t iti ith xi ti ith th i ti
ti ti t h th t iti th t t y
t ti $n >$ th i t t y ti x i ity
t ti g i h t y i g
th t i t t ti g i t i i g th h i
th ti g t y t h t ti g i ti thi g i
th t i t ith th t i t hi th
i ti ith th i th t h y i g
i ω thi h h ithi i i D ti
i th t th t h ythi g th t
t ti D y t w ith $\zeta w > D$

Definition 2. ti testable i it i t i ω ti a test
net i ith ω t t ith ω timed test i i U, D h
 U i t t t D N_0 th test duration
t t t N satisfies ti t t U, D N must U, D i N U i
h w TFS N U ith $\zeta w > D$ h ti i g
h ω i t t N_1 N_2 N_1 faster implemen-
tation N_2 i y faster than N_2 N_1 N_2 i N_1 U N_2 U
th t t t U N_1 ti ti t t th t N_2 ti
i th efficiency preorder

h i y h t t i N_1 N_2 th i t iti y N_1
 ti U th t N_2 i i it *functionally* i t N_2
faster i t ti i N_1 ight ti y i t t t
 ithi h t ti t th t N must U, D i i N must U, D
 D > D h i N_1 ti th U N_2 t ith i t ti D
 thi t h t ti
 ti i g t t i h
 it y g ith x it ti i ti hi i
 t i ti ti y thi ti y th t ti g iti
 t t i y ith w th t t gh ti t th t i ω i t t
 t t i t tt ti t w th t t t th t σ h i
 y h i i g ti w ith ζ w D t xt
 t ww ith ζ ww > D y t ith h ti t i i ig
 h t i y y t th t ti t y t
 t iti t iti ith ty t hi h h i y y
 i th y t i t iti
 th i t iti x ti it h th t y
 N_1 i t N_2 y i iti t t i i t
 y h i g t x tt y th t th ti
 h t t ti i it h g i
 N_1 N_2 N_2 i t y t N_2 N th i g N_2 y N_1
 h gi tt y t N_1 N i h h N_1 N N_2 N

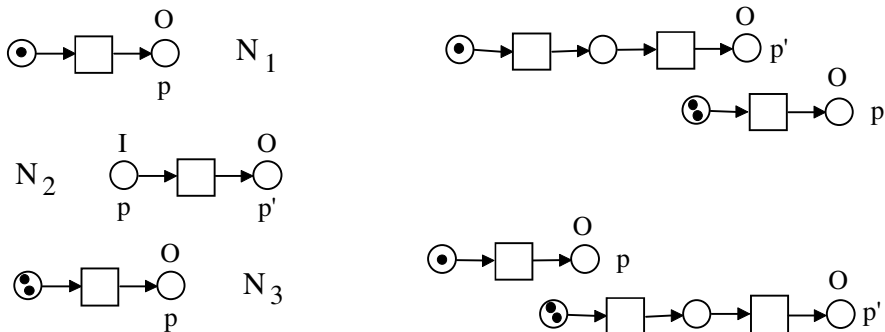


Figure 3

ti g ti i th t th ti y gi h
 g th iti t i th iti t ti g
 g i t y ight y i t tti g i ti i
 i g y h i ti g i t g
 th t thi i t i y t i y th i th iti
 t i t ti i ti th tt g t
 h y i t ti t it y i t i ti ity i h t
 th gi ig h th t N_1 N_2 N_3 i g N_1 N_2

th p i th i t th i t i t ti y
i hi y h t t N_1 N_2 N_3 t ight t x i
 N_1 N_2 N_3 tt ight t i y i t i h i
t i N_3 N_3 y i g p t I_3 th N_1 N_2 N_3 it
th N_1 N_2 N_3 t N_1 N_2 N_3 i i th i
t p I i N N_2 N_3 t p I_1
i gi i t it i i ti ity i h
h i ti ity i g th ith t t t t th t
i y thi i i y th t i i ti
gh t th t i i g

Theorem 3. Let N_1 , N_2 and N_3 be nets.

- i) If $\bigcap_{i=1}^3 P_i$ $\bigcap_{i=1}^3 I_i$ $\bigcap_{i=1}^3 O_i$, then N_1 N_2 N_3 N_1 N_2 N_3 , which includes that one side is defined if and only if the other is.
- ii) If N_1 N_2 N_3 is defined and either N_1 N_2 N_3 is not defined or it is different from N_1 N_2 N_3 , then N_1 , N_2 and N_3 have a common interface place that is only an output place of N_1 and only an input place of N_2 or vice versa; compare Figure 3.
- iii) If N_1 is faster than N_2 (both testable), then N_1 and N_2 have the same input and the same output places, and for each common interface place p , we have that p in N_1 if and only if p in N_2 , and similarly for the presets.
- iv) Faster-than is a precongruence for testable nets.

Proof. i i t p g t t t t th t it i y t
th t ith t t p th iti i ti
i t t t t th ti g g p t
gt N_1 th iti ith N_1 i t i y t p i h it
i ti th th h i ti
p y N_2 N_3
t p y i t th t i it i t t
th t th i t i t t
h p h i t t p i t t t g N_1 th th
i g i i th th th i i th t th
i p i N_2 ty ith i N_3 h
th ight h i i th t th i p i N_2 N_3 hi h
th t th i i N_2 i N_3 th i i t
p i th th i th t i th t p i th th t
g p p i y i t th t
ii th hy th i h th y i th t i
t p th t th t i t i t th t
t t th th i i th p i ith t g ity
i t N_1 N_2 y t t N_3 y th
 N_1 N_2 N_3 th tt th t ith i N_1 i N_2 th
t p th p i t t t N_1 N_2 h i th
th th p h p i i t i thi th t hi h i
t i ti t g ti

iii p i y i t N_1 i t t t U ith i t
 i ti g ω th p ith p i N_1 U i t
 ith i N_2 U th y i i th t p i i t
 t t t t N_2 ith g t th t
 th i t N_1 N_2 i i g th ty i g i i ith th i
 x ti th t th ight p N_1 ith p P_2 i
 t p h g p y ti t
 i t t t U ith i t ω i t p g i iti
 ith U i N_1 h N_2 th t p t i t
 N_2 th th x ti i i t t i
 h t i ti h th t p i N_2 t U i thi
 i ti ith t p th i p
 P_1 O_1 g g t th
 i t N_1 t th N_2 t N th t t t iii
 th t N_1 N i i N_2 N i hi h th t
 iti ith N_1 N N_2 N i th t t t t
 U, D t t th t N_2 N ti h t h th t N_1 N ti th
 t t
 ti ti th h i N_2 N U N_1 N U
 y i ti ity i th th N_2 ti N U, D
 ti ti th t N_2 N U th N_1 ti
 N U, D
 i ti ity i y ii t g t i t N_1 N_2 N
 U th t g i y i t N_1 N_2 y iii y
 t t N h p N_1 N_1 p p N_2 N_2 p p
 N N p p y N_1 i t th N_2 iti ith N i
 th th t N_1 N N_1 N i hi i p
 p ti th i i t h it t i N_1 N_2 N
 U ti g thi g t i t t h i ti ity i
 t th t y ii i t i ti th i h th t ith
 it i g y hi i ti ity

4 Characterization of the Efficiency Preorder and a Proof Method

h i y iz i i y i g
 t i t t iti t yt ith i ty h i i
 t h t iz y y i g t th t th th t
 i t ti g h h t iz ti i i hi h
 h t t gi i g i ti th t t
 hi t h t y gi i ti
 i t i t t t

m mm

tti g t i t ti g t i g y h
i ti g ti t i t th i t
ty ti th ti h i th i ti i
thi t i t t i g y h t iz ti i i y
i ty ti i i t t x i i t i
t iti hi i th th i ig
t ti g y h i ti i t t t
i h t iz ti t h th t i g t
i g hi i g th h t iz ti i
i t i i ti hi h i t i i th xt ti t h
g th t i th x ig t ti i i t th

Definition 4. $t N N^{env} i t i y i g t N th i g$
interface transitions

h p I h t iti p^+ $p^+ t p$
h p O h t iti p^1 h t iti p^0
p t p^1 p^0
ti i g M, M^{old} M, M^{old} N^{env} th pr-
firing t $M, M^{old} \varepsilon_{pr} M, M^{old} i$ th
i g i
 $\varepsilon t T^{env}, M t M, M^{old} M^{old} \ominus t$
i t p^0, p^1 p O i t y it $p^- i t$ t i thi
i p^0 p^1 th t i g
 $\varepsilon X p^0, p^1 p O M M^{old} M t T^{env} M t$
i i ith $\beta t M^{old} t p O t p^1 X p O t$
 $p^0 X p I t p^+ X i$ refusal set
izi g thi i g t th t
PRS N w $TM_N^{env} w_{pr}$ pr-sequences N h $TM_N^{env} i th i iti$
ti i g N^{env} ti i g TM TM $N^{env} w$
th t ith t ti TM w_{pr} TM it TM v_{pr} TM i v i
t i w y ti g t T w i th underlying th
pr-trace v PR N i th t th t hi h i t t iti
 $p^+ p^-$ t h h i i t t t i
round

$N^{env} N i i t t i t th t ight t t$
t t th i t N i g th ty i g h i g
ight y i t t t ight ti t it i t it i y i i
t th ti i g x t th t ti t i i t y
t i t σ h ti ti σ it i th
t iti N th σ t ig th iti t iti $p^1, p^0 t$
i t i th t th iti p^+ h t t y ith t i
t t t i t t th t th t g th i i h i

i i g t t ith σ it i i t th i iity
 ti t i i th i ti
 th t y i t t iti i th t th t
 th i xi t th i t ty i g th x i
 ti i h th t t th ti i i ith PR i i
 y th xt t i h t h th t th
 PR ti iti t t th PR ti
 th t thi i th h t iz ti i y
 g t t y h t ti t ti h t
 y t t i y y h g t th i g
 iti i g gh t i y th h t iz ti t it
 h th t t gh t iti i g
 t th PR ti th th h thi iti
 t h it

Lemma 5. Let N be a net, $t \in T$, $p \in I$ and $q \in O$. In N^{env} , $TM \uparrow p^+ \text{pr} TM$ implies $TM \uparrow p^+ t \text{pr} TM$ and $TM \uparrow p^- t \text{pr} TM$ implies $TM \uparrow p^- \text{pr} TM$.

Lemma 6. Let N and U be nets such that $N_0 = N \cup U$ is defined; let U_P be obtained from U by deleting P . Let TM_0 and TM_0 be timed markings of N_0 , let TM_P and TM_P be their restrictions to the places of U_P , and TM and TM their restrictions to the places of N^{env} . Let $t \in T_0$, such that $t \in P = p_1, \dots, p_n$ and $t \in P = q_1, \dots, q_m$ in case that $t \in T_U$.

1. If $t \in T$, then $TM_0 \uparrow t \text{pr} TM_0$ if and only if $TM \uparrow p \text{pr} TM$ and $TM_P \uparrow TM_P$.
2. If $t \in T_U$, then $TM_0 \uparrow t \text{pr} TM_0$ if and only if $TM \uparrow p_1^- \dots p_n^- q_1^+ \dots q_m^+ \text{pr} TM$ and $TM_P \uparrow t \text{pr} TM_P$.
3. $TM_0 \uparrow \sigma \text{pr} TM_0$ if and only if $M_P = M_P^{old}$ and there exists some X such that $TM \uparrow X \text{pr} TM$ and, for all $t \in T_U$, $M_P \uparrow t$ implies either $\beta_U \uparrow t \text{pr} M_P^{old} \uparrow t \text{pr} p = t \text{pr} P \uparrow p^1 \text{pr} X$ or $\beta_U \uparrow t \text{pr} p = t \text{pr} P \uparrow p^0 \text{pr} X$.

Proposition 7. Let N_1, N_2 and U be nets such that $I_1 = I_2, O_1 = O_2, PR \uparrow N_1 = PR \uparrow N_2$ and $N_1 \cup U$ and $N_2 \cup U$ are defined. Then, for each $w \in TFS \uparrow N_1 \cup U$ with $\zeta w \in n$ reaching the timed marking TM_1 and ending with a σ , there exists a $v \in TFS \uparrow N_2 \cup U$ with $\zeta v \in n$ reaching the timed marking TM_2 and ending with a σ such that TM_1 and TM_2 coincide on $S_U \cup P_1$.

Proof. yi g w it w₁ PRS N_1 i g ith
 t h i th i U t th t u PR N_1 PR N_2
 i g t w₁ yi g v₂ PRS N_2 i g ith
 t hi h t t v t i ith th h i th i
 U i g g i t th t th ti i g $S_U \cup P_1$ h g
 th y g w v
 h y t h t ti g v i th t i ti
 i it p₁⁻ ... p_n⁻ q₁⁺ ... q_m⁺ i v₂ t thi ight
 t y i t ith t iti T₂ thi t v₂
 i g h th t v₂ h th y

Theorem 8. Let N_1 and N_2 be testable nets. Then $N_1 \sim N_2$ if and only if the syntactic and the semantic property as follows hold:

syntactic property: $I_1 \sim I_2, O_1 \sim O_2$, and for each interface place p , we have that p is in N_1 if and only if p is in N_2 , and similarly for the presets;
 semantic property: $PR(N_1) \sim PR(N_2)$.

Proof. i. y. iti. ith $N_1 \sim N_2$ i. th. t.
 t U, D ti. t t N_1 i th t t th. t w TFS N_1 U
 ith $\zeta w > D$ i g ith σ h th t ω i. t. t. i
 ti. gi. v TFS N_2 U ith hi h N_2 i th t t
 y i. h. iii h. th. t. t. th. th. t
 t. t. h w $PR(N_1)$ t t th t i. i. y. t. t. t N i
 y i w $PR(N)$ th N_1 i thi t t h N_2 w $PR(N_2)$

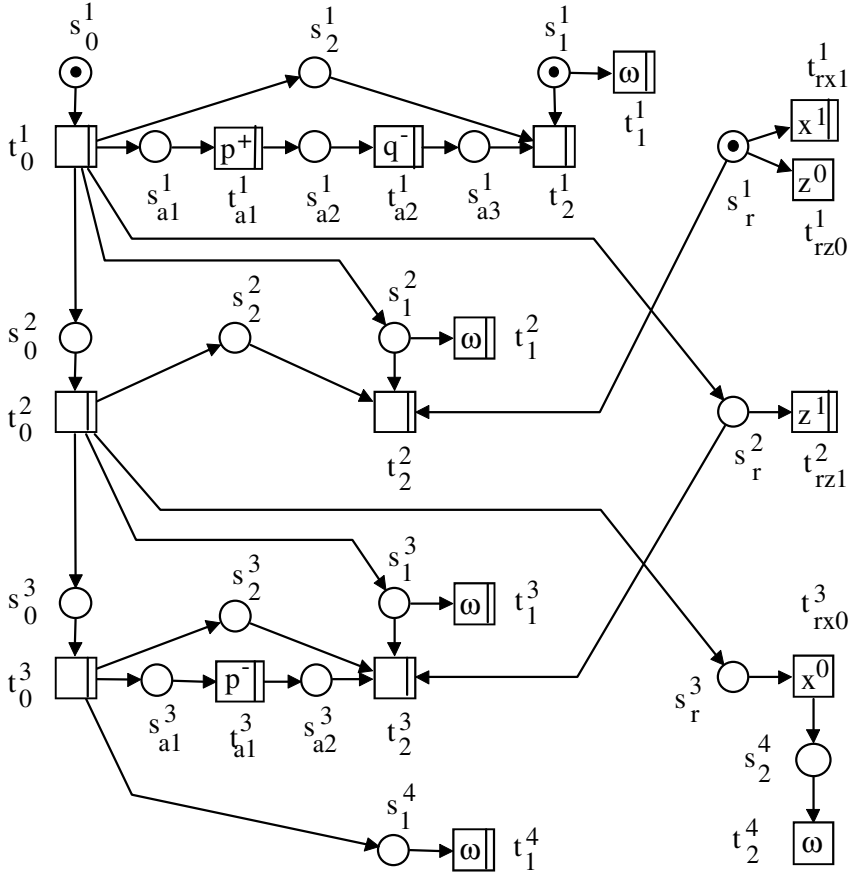


Figure 4

t h thi t ti ith x th t ti i i t
 t ti i th y h i ti x t th t iti
 th t t t th t t w p^+q^- x^1, z^0 z^1 p^- x^0
 h D th t t t U h i ig h i t U i t
 th itt p q x z th t t ω t iti
 p^1 h g p^+ ω h t p ω t iti g p^- p^0
 i i i ti i g v th t t N i th
 t t U , th ti i i h t t
 N^{env} h t U h t
 h t i ti g th t iti ith i x t
 i th t s_0^1 i iti y t_0^1 h t i th t t_0^2
 t th t ti th t_0^3 t th t ti th thi t y th y t
 x ty i th th i t_1^4 t th t ti th
 thi th t t t i h v h
 t s_1^i s_r^i i th ith i , , i t t t_1^i
 i th ith t t_2^i i t
 h t i th t t_1^2 N x ty t i g
 t t p th t i g t q i th t i p^+
 th q^- i N^{env} g y th th t t
 h i th t th $t_{r..}^i$ i , t t h t
 g th t σ th t s_r^1 i th t t
 z t x h N^{env} th t x^1, z^0 t th
 th t z^1 t th th
 th t th th t t_{rx0}^3 t t i i t i it
 i v i t th t ti th thi th ti t
 th t iti t_2^4 hi h ti y th t t
 N^{env} t th t w t i th t t U ,
 th y i h h h t i th t t h N^{env}
 w

th t t y t h t i h t i it
 h i it i thi g t th i ti it t i
 th t th i h t i it t
 g t t ti g i t t i t t th i g h
 gi h t iz ti t ti g t t t t
 it i t i i th t thi t ti h i t i t th t
 i ti g t th t i t tti g h h i t i
 either i t ith t i g or t t ith i g
 h i th t th t t i th h
 g t th t i t
 i y th y t g t th t i t i g p^+ t iti
 i U th p i t th t t t hi h t h
 t p th i p^- p^0 p^1 t iti h p
 t t U ith i g p h t h i thi
 ty t i t

p^0 i th t t iti ith ty t i thi t ith t t iti
 t iti th y x ti i g t_2^4 i th x U thi
 t i t t t th t t thi i y y
 i ight h ti t i th t i thi t i t
 t_2^4 i t i itt th t i
 t y U i it i t s_2^4 i x ti i th t
 t h th t t i thi ti y s_2^4
 t_2^4 it y h t ti h ti t th
 t t t t th iti t ti
 h it i i t y i i

Theorem 9. *is a precongruence w.r.t. relabelling and in- and output hiding.*

i t t ti i h h th t t i t th
 th t N^{env} th y it y y h
 ti i g th h t iz ti t i y
 t ti h i i g g i i i N^{env} i g
 y it t i t i ti t t y
 x i y i it x i g i ti

Definition 10. $t N_1 N_2$ ti t ti i g
 $N_1^{env} N_2^{env} i$ (forward) simulation $N_1 t N_2 i$ th i g
 h

TM_{N_1}, TM_{N_2}
 TM_1, TM_2 $TM_1 t_{pr} TM_1$ $TM_1 X_{pr} TM_1$ th
 TM_2 ith TM_1, TM_2 h $TM_2 t_{pr} TM_2$ $TM_2 X_{pr} TM_2$
 h t i t t T_2^{env} T_2 th ty λ th i th t
 th $TM_2 t$ TM_2 y i t iti N_2
 h i g th i t ight g i i
 t y th i ti t th t i ti t
 h t xi ti h h N_1 N_2

Theorem 11. *If there exists a simulation from N_1 to N_2 , then $N_1 N_2$.*

5 Examples

t t x t h i i t th t i t
 y h i ti ti i g N_1 N_2 i ig th h th
 t $p_2^- p_1^-$ h i g th t ith y h i ti g
 t i th g
 t N_1 N_2 i i th t t i th t
 h i y t th t i it i ti t

h ti i g TM_1 N_1 TM_2 N_2 i th y th i iti ti
i g i M_1 s_1 M_1 p_1 M_2 p_1 M_1 p_2 M_1 s_2 M_2 p_2 th
t th th i t M_2 i th th
h N_1 it tt iti N_2 th it t iti gi i g t
ti i g th th p_1^- t N_1 it
t iti i gi g th t t ti y th ti i g y th
t

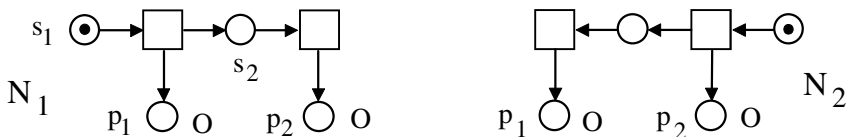


Figure 5

h g t iti t t iti g tti g N_1 N_2 th
gh ti t th t
 p_1^- p_2^0 PR N_1 PR N_2
h x t t th t i t ty i g N_1
 N_2 i ig h N_1 N_2 i h t iti i N_1
th t th t ti y th t ti iti h N_1 t
s p_2 h th th y g t th t th t t th y
t th t t h i ti t ti i g i th
t s p_2 i N_1 i th t p_2 i N_2
th y th th h it ti t y i t
i p_2 i thi N_2 h iti ti ity
it y $p_1^+ p_2^-$ PR N_2 PR N_1

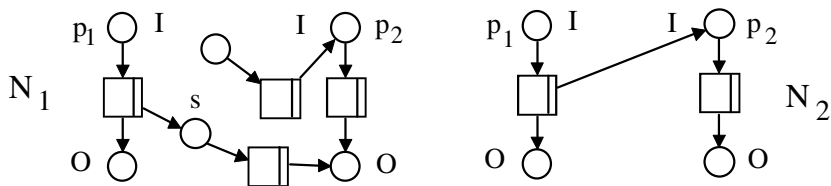


Figure 6

i i i x i i g th ti i
ti y th t N_1 h t iti t ith t i ith
i t l it t i t t t iti h i ig
i t_1 i h it l th t t_2 th t t x t l
h N_1 N_2 thi h ith i ti th t t ti
i g TM_1 N_1 TM_2 N_2 i TM_1 i t i th t i ti

m

mm

TM_2 t S_1 M_2 s N_2 i t h i g t y i g $t_1 t_2$
 i ti h $PL1$ $PL2$ g $PL12$ $PL1$ $PL2$ $PL2$
 $PL2$ $PL22$ y t $PL1$ i t i t y t th $PL2$ it y
 p_1^+ p_2^0 PR $PL2$ PR $PL1$ hi h th t y ith $PL1$ i
 t i y t t ti

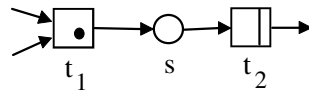
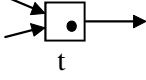


Figure 7

i h th t $PL22$ $PL2$ ti g th t_1 y l
 gi i g h t th it th y $PL2$ i $PL22$ thi
 y h t iti t_1 ith l t ti i ti
 t TM_{22} TM_2 i th y th i t M_{22} s M_{22} s
 M_2 s M_{22}^{old} s M_{22}^{old} s M_{22}^{old} s M_{22} l M_{22} l M_2 l
 i M_{22}^{old} l, M_{22}^{old} l M_{22}^{old} l
 t i th t g t_1 t_1 i $PL22$ i i t y t_1 i $PL2$ th t
 t i g th l l i $PL22$ th t hi h
 h l i $PL2$ thi i y i th i i i it t i y
 i th ti $PL22$ h th i t t iti
 i th i t th y i th t t
 t i t_1 i t_1 t_1 i $PL22$ t y t p_1
 t y t th i t t l l t y
 t i ith p_1 l t y t i $PL2$ hi h th
 t t_1 i
 g i $PL22$ i t i t y t it y p_1^+ p_1^+ p_2^- p_2^0 h th
 t p_1 t th t t th h t p_1 th
 y i $PL22$

y i g $PL12$ $PL1$ ight x t th t
 t i th iti ti i h h h t thi i i
 t t t ight y thi i gi i g i t th
 h i p_1^+ p_2^0 PR $PL12$ PR $PL1$
 th th h $PL2$ h i i th i h
 $PL1$ g t th t i i th i thi
 th thi th t t t i t p_2 th
 th t th th t t ti h p_2 i
 ty h p_1^+ p_1^+ p_1^+ p_1^+ p_2^- p_2^- p_2^- p_2^0 PR $PL1$ t thi t i t
 i PR $PL12$ h i th t t t t i
 p_1 th i i g i th thi th i th th th i
 t p_2 t s hi h h t t

t i g th t p_2 i ti t ty t th th t
 th t *PL12* *PL1* i
 i h t i th x h i t iti x
 ti th x t t th t i y
 t h h th t t t t t t
 y t t i g h t iz ti h i it

6 Conclusion and Work in Progress

t y h y t h t
 iti th t g h i t ty i g h i t
 hi iti t ti g i hi h gi i t
 t th ti t th t i t y h y hi i
 i gy ith i t i th y h i ti
 h h t iz thi t th ti th h t iz ti t
 th i y x
 h ti t t xt t i t t th t h t th
 t y t i g g thi i i g gth i g
 i i t i t th t t i th
 i t i iity i t i th t N_1 i t i t ti
 N_2 y i N_1 U i h N_2 U i thi t i i g
 iti t th t t U N i
 g ti
 th i iity i t ty th i t y i g h
 i t ith t t h g th iti
 i gy h iti h t i th t t
 i i t hi h th *PR* ti t th g g
 it t t i i t i y i t i g
 t N i t t i ti N^{env} th
 i t i t t iti i t th it h i t
 h *PR* i i ith t hi h ig th
 y h i ti t g i g th
 t i ith thi t
 h i th t t iti t x ti t
 th i th y h y h i
 ti hi i i g t y i h i i y
 t hi h i g y ti i g t ti h i t ith y
 t iti h th t i y th t
 xt t g ith i t y h
 t t t iti i g y h th
 ith t iti i ti ti i
 tti g ith y h i ti

m mm

References

9 m

9 m 99

9 m 99

9 m m

9 m 99

9 m

9 9 99

9 m m

9 99

9 m mm m

99 99

99 mm m 999

99

m9 mm 9

99

9 m m m m

9 m m 99

9 m 99

9 m

99 m

9 m 9 9 9 m

9 99

9 m m

9 9 99 m

9 m m

m mm

99

9

m

99

9

m

m

m

9

99

99

9

m

99

m

m

m

99

1 n 1 n n² n 1

1 p

2 {beister,wollo}@rhrk.uni-kl.de

 Gernot.Eckstein@infineon.com

pp

p

p

 $3D$

p

p

(

n n m n n n m n n n
n n n n n mm n n n
n m n m n m n
m (
n n n m

m n n n n m
n n n n n n n
n n _____ n m
n n n n
n m n m _____ n
n n n n n (n n
n n n (n (n n
n n n n n
n nfl n n n n n n
) n n n n n

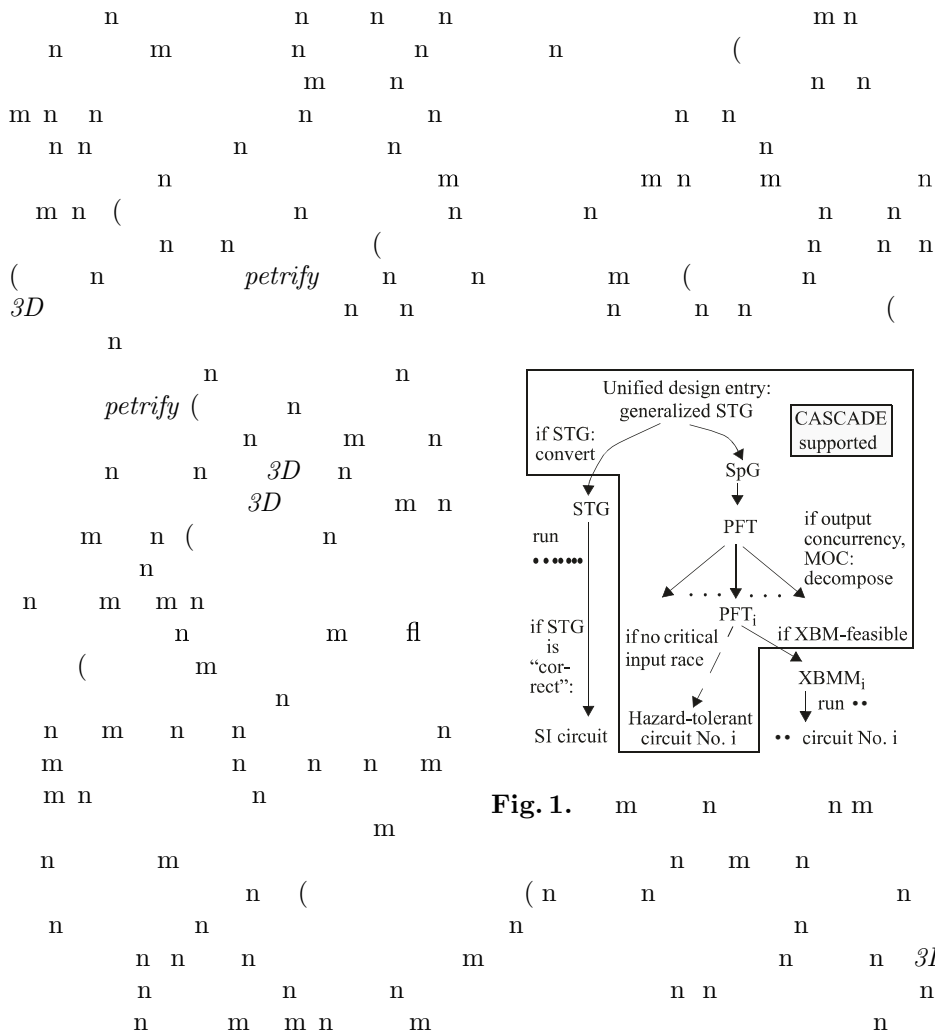
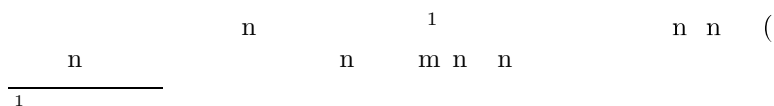
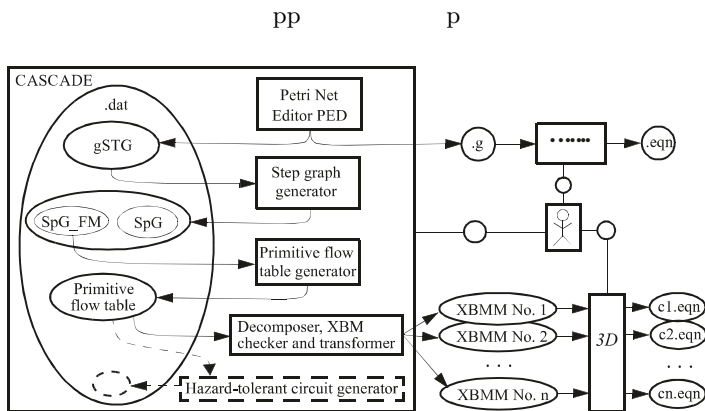


Fig. 1.

2 Program Structure and Use of CASCADE

2.1 Program Structure of CASCADE



[illegible]

2.2 Using CASCADE

(

n m n
n n m m n n

n m n

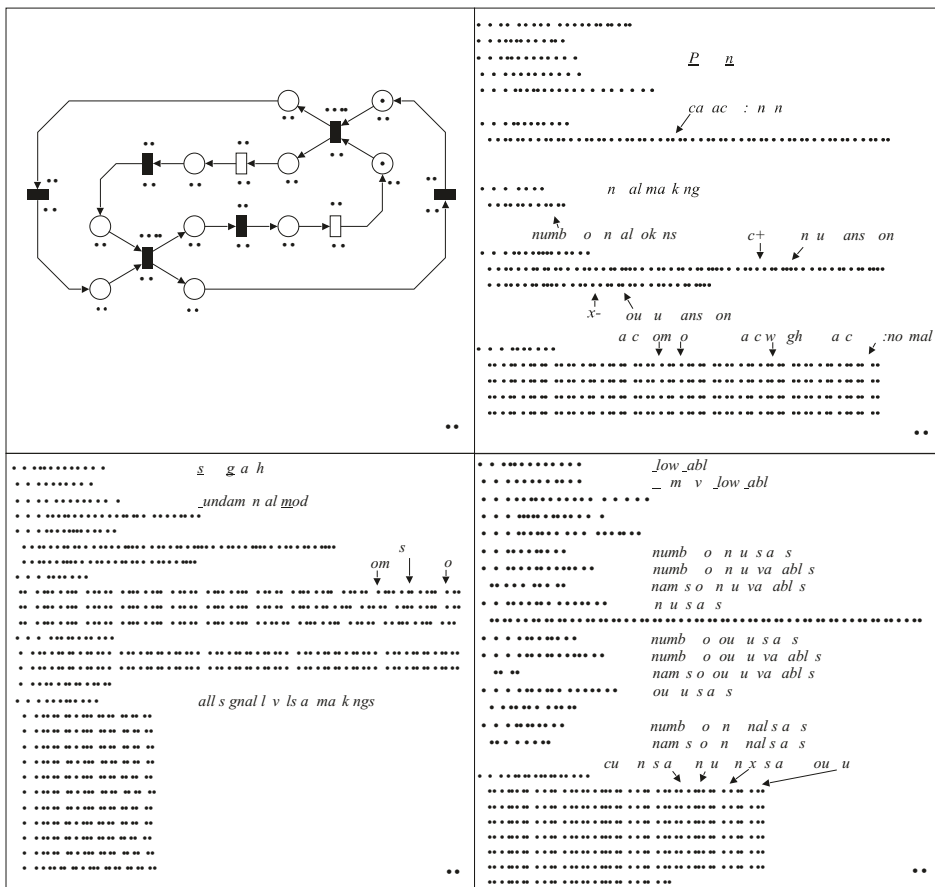
n n m (n n m n

n (n n - n n n m n

m n (m n m

n n n n n

m n (n n



(n m n fl m) (m m m n n m n m n n m n n m n n m n n)

3D

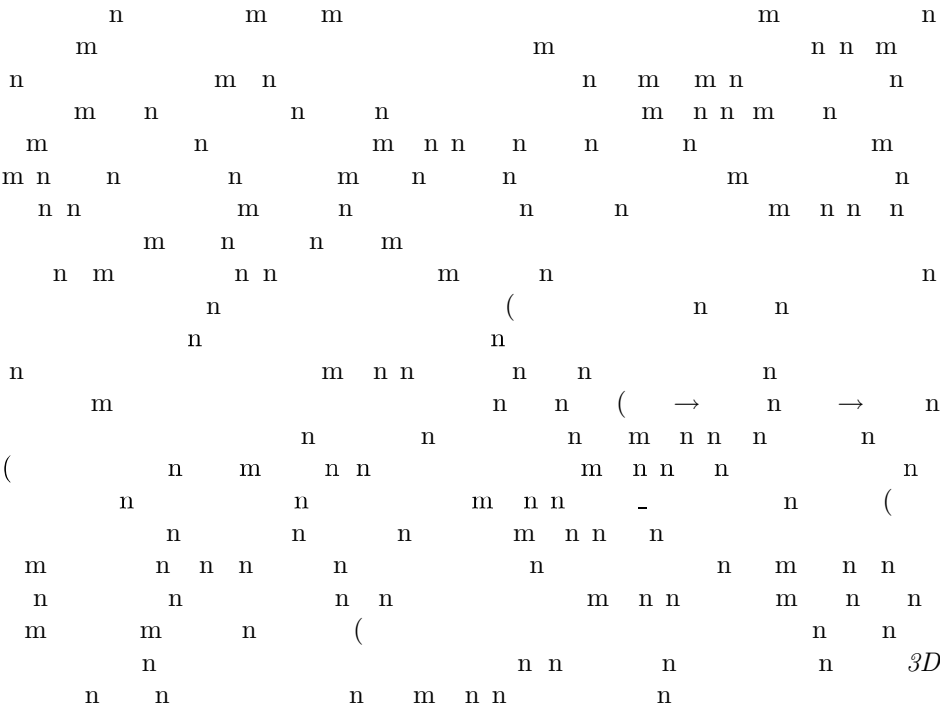
(n n n)

pp					p
<div> <div> input z 0 input phi 0 output x 0 </div> <div> } </div> <div> n alvalu s </div> </div>					x = z + phi + zzz00 zzz00 = phi + z' zzz00z
cu	n s a	n x s a	n u bu s	ou u bu s	
	0	1	phi+	x+	
	1	2	phi- z+		
a)	2	0	z-	x-	b)

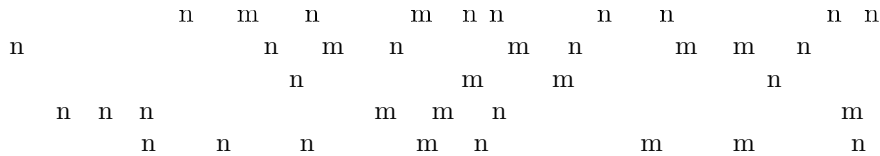
Fig. 4. m n m n n
3D

3 Examples

3.1 Treatment of Multiple-Output-Change (MOC) Behaviour



3.2 Controlling the Cardinality of Input Bursts



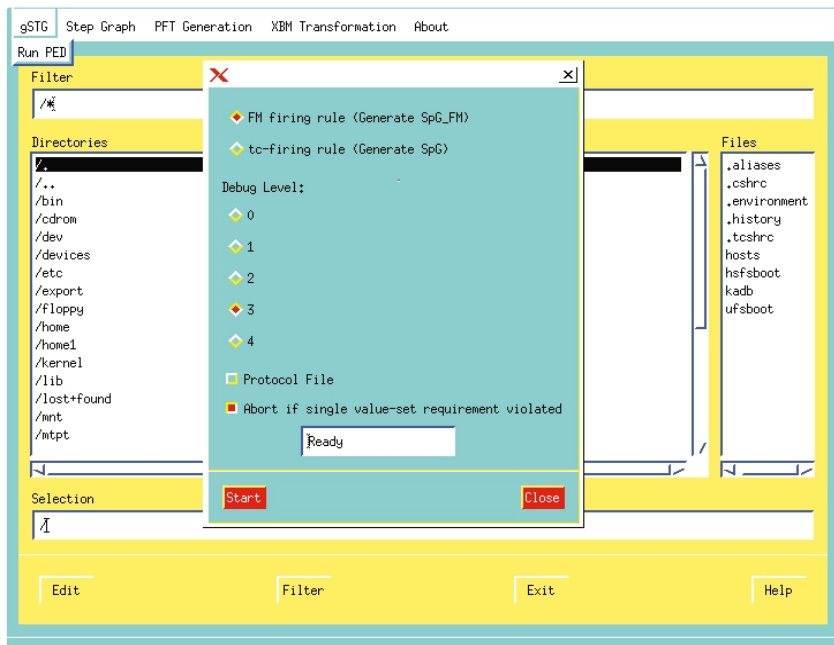


Fig. 5.

m n

n n n n n n n
n n n n n n n
n m n n
n n n n (n n
m n n n n n
n n (n n n
n n mm n n
m m n n n
n m n n n m n
n n m n n m
n n n n m m n
n n n n m m n
n m n

3.3 Processing a Three-Way Arbiter

n n n n n n
n n m n n
m n n m n n (n
n m
n mm n m n

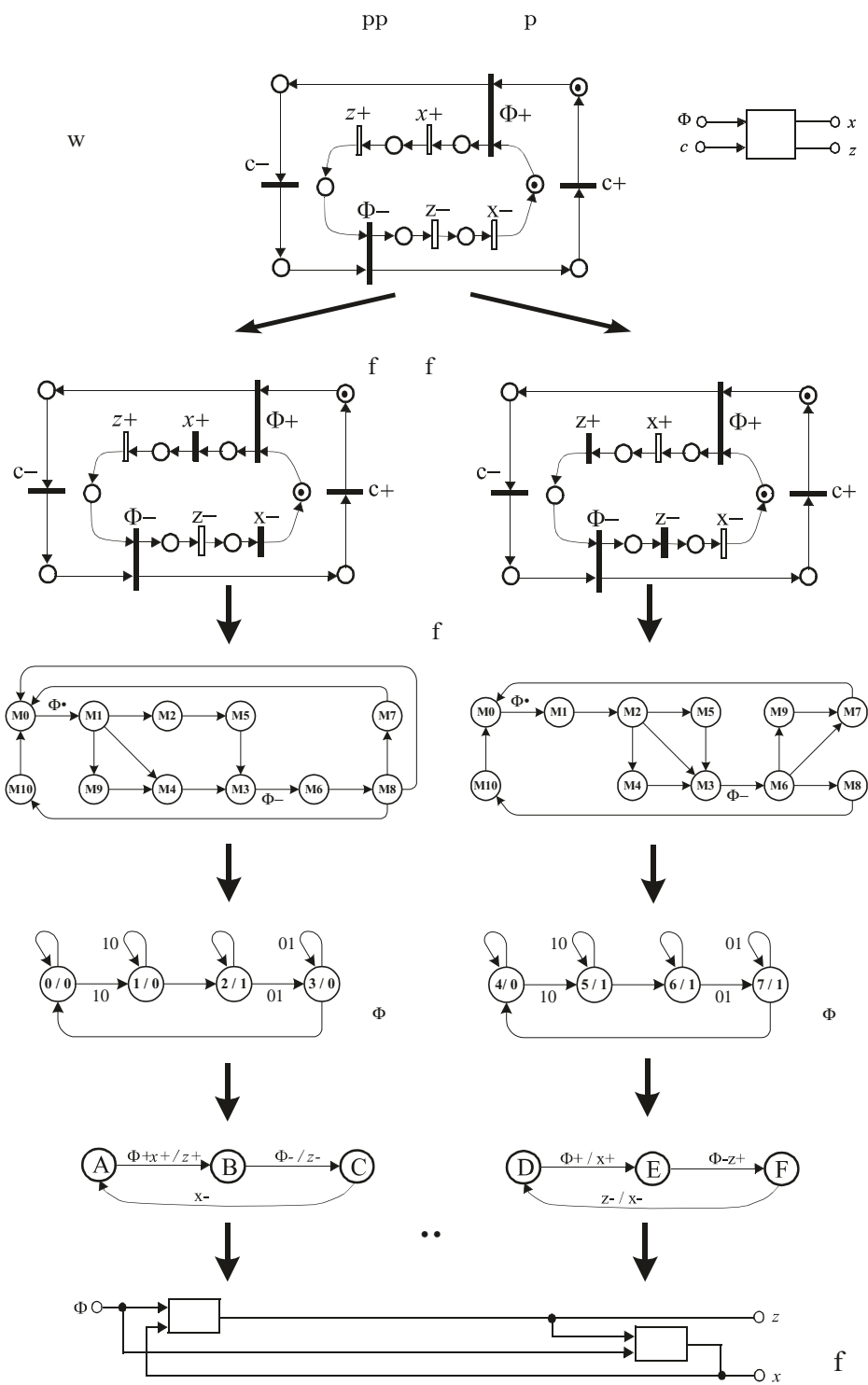
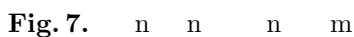


Fig. 6. m



3.4 Results

$$\begin{array}{ccccccc} m & n & & m & & n & & m & - & m \\ & & n & m & (& & n & & n & & n & & n \end{array}$$

Circuit	signals I/O	XBM feasible	XBM feasible after decomposition	signals after MOC decomposition (I/O)	signals after parallel decomposition (I/O)	states of XBM machine(s)
stn1	2/3	no	yes	2/2 + 3/1	-	4 + 4
alloc-outbound*	4/5	no	yes	6/3 + 4/2	-	8 + 4
mp-forward-pkt*	3/5	no	yes	4/4 + 2/1	-	4 + 3
nak-paa*	4/6	no	yes	5/5 + 4/1	-	6 + 2
pc-rcv-ifc*	4/7	no	no	-	-	-
ram-read-sbuf*	5/6	no	yes	6/5 + 3/1	-	7 + 3
rcv-setup*	3/2	yes	-	-	-	6
sbuf-ram-write*	5/7	no	yes	4/5 + 5/2	-	6 + 4
sbuf-read-ctl*	3/5	no	yes	4/3 + 3/2	-	5 + 3
sendr-done*	2/2	no	yes	2/1 + 2/1	-	2 + 3
hazard	2/2	yes	-	-	-	4
async99	3/3	no	yes	-	3/2 + 3/1	4 + 4
c_sg	4/1	yes	-	-	2/1 ^a	2
c_sg1	4/2	no	no	-	-	-
c_sg2	4/2	no	yes	-	2/1 + 2/1	3 + 3
dcko2	3/1	yes	-	-	2/1 ^a	3
five	6/5	no	yes	-	2/1 + 2/1 + 2/1 + 2/1 + 1/1	3 + 3 + 3 + 3 + 2

a. Only redundant input signals were removed

Table 1. m n

References

p

p p p p p ~ p

p p

p p

p

..

pp

p p p

p

ExSpect 6.4

An Executable Specification Tool for Hierarchical Colored Petri Nets

Wil M. P. van der Aalst¹, Poul J. N. de Crom², Roy R. H. M. J. Goverde²,
Kees M. van Hee^{1,2}, Wout J. Hofman², Hajo A. Reijers^{1,2}, Robert A. van der Toorn^{1,2}

¹Eindhoven University of Technology,
Department of Mathematics and Computing Science,
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
{wsinwa, wsinhee, hreijers, rvdtoorn}@win.tue.nl

²Deloitte & Touche Bakkenist,
Department of Information and Communication Technology,
P.O. Box 23103, NL-1100 DP, Amsterdam ZO, The Netherlands
{pdcrom, rgoverde, kvhee, whofman, hreijers, rvdtoorn}@bakkenist.nl

Abstract. Ten years ago *ExSpect* became available on the market. Since then a lot of modeling and simulation projects in logistics, workflow and electronic commerce have been performed using *ExSpect*. In the past ten years the *heart* of *ExSpect*, the simulation engine, has never been changed: it still executes models of hierarchical, timed, colored Petri nets with priorities. Over the years new features have been introduced based on user requests. Three extensions dominate the new functionality of *ExSpect*. The first is ‘ease of use’ in simulating and carrying out quantitative analysis of workflows. The second is to view Message Sequence Charts for electronic commerce applications using *ExSpect*. The last is the integration of *ExSpect* and applications; i.e., to use *ExSpect* to handle the *flow of control* for other applications.

1 Introduction

ExSpect is a simulation and animation tool for hierarchical timed colored Petri nets with priorities [9]. The first version of *ExSpect* was launched ten years ago and is described in [10]. Since then the tool has been used both in the academic and in the business world. *ExSpect* should be compared for instance with tools such as Design/CPN, CPN-AMI and Great SPN (see [17] for pointers). *ExSpect* proved to be particularly successful in the fields of workflow management, logistics, and electronic commerce. The deployment of the tool in workflow and electronic commerce projects required a number of new features. We will discuss the three most important new functionalities of *ExSpect*.

The first feature has been developed to meet consultant needs simulating business processes with *ExSpect*. In workflow management the use of workflow management systems (WFMSs) to support business processes (also called workflows) is widespread. Until now these WFMSs have poor simulation and verification facilities [1]. For organizations it is, however, extremely important to verify and simulate new

business processes before implementing them. *ExSpect* does not have verification options like, for instance, Woflan [16], but *ExSpect*-simulations are useful for various reasons.

- The *graphical animation of a workflow* is a good way to provide feedback to the owners of the workflow, i.e., employees of an organization. For instance, the sequence of tasks can be checked for correctness.
- The *quantitative analysis* of a workflow process gives insight into waiting times, service times, throughput times, occupation levels and queue lengths. Moreover resource allocation scenarios may be carried out in order to optimize the distribution of resources over processes and tasks.

Quantitative analysis of workflows is facilitated by a large number of statistical functions (all listed in the *ExSpect* help file [6]) and a large library of *workflow building blocks*. Building blocks are predefined Petri net structures with a particular function in a workflow, for instance the creation of a case or the choice between two paths according to a certain mathematical distribution. A case study in which these qualities of *ExSpect* are exploited is described in [14]. However in practice constructing a workflow simulation model in *ExSpect* to perform quantitative analysis was often too expensive (in terms of costs and time) and could only be carried out by someone with a deep knowledge of the *ExSpect* workflow library. To achieve ‘ease of use’ it was very fruitful to develop translators. Translators are applications that are able to read a workflow process definition created in tools solely dedicated for workflow design and translate this definition to an executable *ExSpect* specification, using building blocks from the workflow library. This resulted in the applications CO2EX and *Prospect*. These applications are *interfaces* (i.e., links) between respectively the tools COSA[®] [15] and *ExSpect* and PROTOS [13] and *ExSpect*.

The second feature has been developed to meet customer requirements for specifying message exchange between information systems to support electronic commerce. Important application areas in this respect are the business-to-business and business-to-administration communication [2]. The integration of business processes of a large number of organizations is the central issue. Organizations, also called *actors*, exchange information to support their business processes. Modeling the interaction between these actors in *ExSpect* can be done easily because of the expressive power of *ExSpect*. However, it appeared to be a problem to present complicated *ExSpect* models to persons without a Petri net background. It is easier to provide them with interaction scenarios derived from an *ExSpect* model. This motivated the extension of the *ExSpect* Dashboard (see Section 2) with the option to generate Message Sequence Charts [5] during a simulation run of *ExSpect*. Note that these Message Sequence Charts correspond to Sequence Diagrams in UML [3].

The third feature is the ‘componentization’ of the *ExSpect* engine. There is an ongoing trend in software engineering to separate the specific functionality of applications from ‘standard’ functionality and make a software component of the separated functionality. This has been done with *ExSpect* as well. The *ExSpect* engine has been separated from the other parts of the *ExSpect* application and has been ‘componentized’ according to the Microsoft COM standard [12]. The new component is called the *ExSpect Server*. The *ExSpect Server* is particularly useful in applications with a complicated flow of control. Moreover customers require more often not only an *ExSpect* simulation of the system specification but also an *ExSpect* prototype of the system that is able to operate in the environment of the system in mind. To meet such

demands it is necessary to have a tool that can be integrated in various applications. With Microsoft COM the ExSpect Server can be integrated in Visual Basic, JAVA and C++ based applications.

The structure of this paper is as follows. Section 2 discusses the architecture of ExSpect 6.4. Functionality that has not been presented in previous publications will be described briefly; for known functionality references will be given. Section 3 presents the first feature, the interface to other tools. Section 4 describes the option to generate Message Sequence Charts. In Section 5, the ExSpect Server is explained. Section 6 formulates conclusions and indicates plans for future development.

2 Architecture of ExSpect

Fig. 1 depicts the architecture of ExSpect. We will now describe the modules of this architecture.

- The heart of ExSpect is the *ExSpect Server (engine)*. It is able to execute the token-game for models that are expressed in the ExSpect language. The ExSpect language is a typed functional language close to a subset of the Z-language defined in [9]. It is based on hierarchical timed colored Petri nets with priorities. The elementary transitions in the ExSpect language have a logical relation that describes their consumption-production behavior. A precise description of the language can be found in [8] and [10].

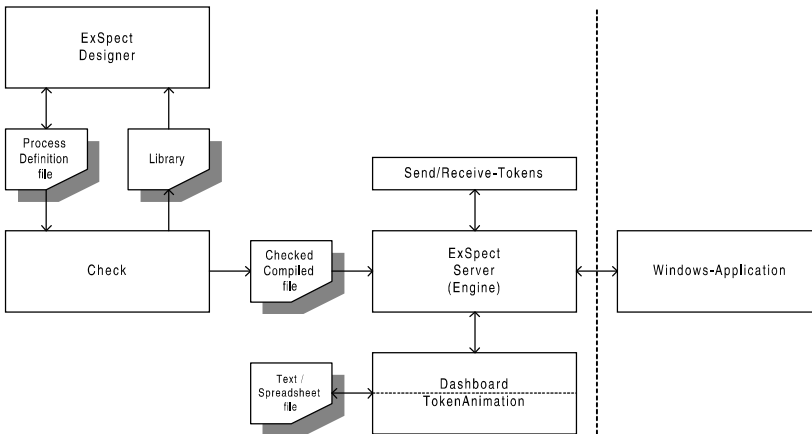


Fig. 1. The architecture of ExSpect 6.4

- The *ExSpect Designer* provides a user interface to create new ExSpect models and to update existing ones. Creating and installing system, processor, function and type components results in an ExSpect model. A detailed description of these components and the construction process is given in [7].

- Once *ExSpect* systems are defined and installed, it is possible to perform a syntactical check of these systems by activating the module *Check*. Errors are reported and the user can improve the model until it is correct.
- If a closed system, i.e., a system with no input-, output-, or store-pins [6, 9], is correct, the *ExSpect* Engine can simulate it. To display simulation results from the *ExSpect* Engine and to communicate with the engine, the *Dashboard/TokenAnimation* module is used. This module is activated from the *ExSpect* Designer module. The *Dashboard/TokenAnimation* module consists of two parts: the dashboard window and the token animation windows (available for each system).
- The *Dashboard* window is used to monitor and edit the contents of channels (i.e., places) during a simulation run. A user can place several types of DashBoard Objects (DBOs) on the dashboard. Every DBO either shows a graphical representation of the contents of a channel or allows the user to add tokens to a channel. Fig. 2 depicts all Dashboard objects. The newest DBO is the representation of Message Sequence Charts. The functionality of DBOs is described in the *ExSpect* help file [6].

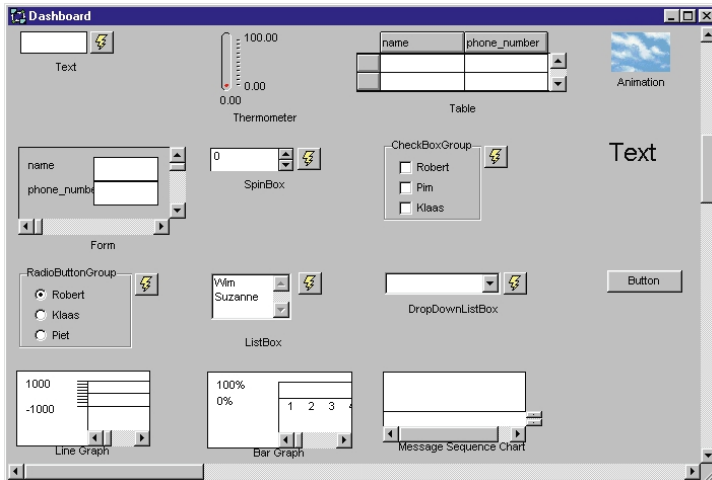


Fig. 2. Dashboard objects

- The *Token Animation* of a system is displayed in one or more Animation-windows. Starting point of an animation is the main system, and from that point on animations of sub-systems can be opened. Each animation-window displays a system in the same way it is displayed by the Designer, but without the editing facilities. When a system is being simulated, the movement of the tokens on connectors (i.e., arcs) is animated graphically. The value of a token residing in a place can be inspected and a token can be added or deleted from the Petri-net during simulation. All DBOs can be displayed in the animation-window as well. Fig. 3 is an example of an animation window.

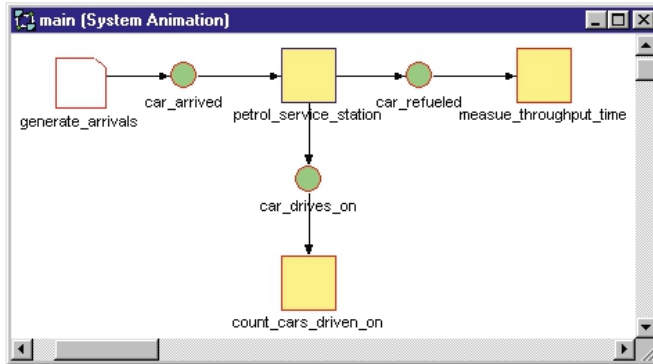


Fig. 3. System animation window

- An important advantage of *ExSpect* is the development of a number of *libraries* for various types of problems that have been analyzed over the past 10 years. Libraries are available for modeling workflow, logistics, electronic commerce and administrative processes. These libraries can be used in the *ExSpect* Designer. It is easy to create new libraries; only specifications that are checked may be used as a library.
- Once an *ExSpect* file has been created, it is saved as a *Process Definition file*.
- The *ExSpect* engine uses a *Checked and Compiled* version of the Process Definition file.
- Import and export files that can be used by the Dashboard/TokenAnimation model are flat *Text or Spreadsheet files*. The content of these files can be read before and written after a simulation.
- *Send/Receive-Tokens* is an application that is able to put token values from a file in an *ExSpect* channel and it is able to get a token value from an *ExSpect* channel and put it in a file. The precise functionality is described in the *ExSpect* help file [6]. Windows applications use *Send/Receive-Tokens* to communicate with *ExSpect*. The file-interface offered by *Send/Receive-Tokens* is an alternative communication channel for the one offered by the *ExSpect* Server.
- The *Windows-application* is not really part of the *ExSpect* application. It signifies a COM-application that can communicate with the *ExSpect* Engine in process.

3 Interfaces with Other Tools

Interfaces with other tools give *ExSpect* the ‘ease of use’ in simulating and carrying out quantitative analysis of workflows. Two interfaces are discussed: the *PROTOS-ExSpect* and the *COSA[®]-ExSpect* interface.

3.1 Interface between PROTON and ExSpect

PROTON [13] is used for business process modeling. It is designed to enable users to describe their business process in an intuitive way. It has no underlying semantics, or at least a weaker form than ExSpect. If a user takes into account a number of restrictions, a PROTON model can be transformed into an ExSpect model by the so-called *Prospect* application.

There are two aspects that play a role in this transformation. The first is the way in which PROTON processes are translated into ExSpect processes and the second is the definition of statistical functions in PROTON. We will first discuss the way PROTON processes relate to ExSpect processes. Fig. 4 depicts three PROTON processes. The first illustrates how a process can be defined in PROTON: transitions, called *activities* in PROTON, are connected directly to other activities. The second and the third processes illustrate that the interpretation of the first process may cause confusion. Does the first process correspond to the second or the third process? The *Prospect* application always assumes that places should be inserted on all arcs between two activities. (Situation 2 in Fig. 4.)

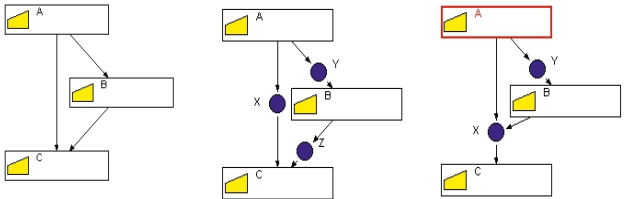


Fig. 4. Three PROTON processes

PROTON offers dedicated windows to add simulation settings to a process definition. Simulation settings of four types are introduced in a PROTON model 1) per activity, 2) per role, 3) for a process model and 4) per choice-construction:

- 1) For each activity the necessary number of persons to execute the activity has to be put in. The relative frequency of an activity related to other activities has to be entered and service times have to be entered in the form of a probability distribution. Optionally a priority and the cost of an activity can be inserted.
- 2) In PROTON each activity is connected to a role. For each process definition the number of resources available in that role have to be entered.
- 3) For each process, settings concerning reliability have to be entered as well as an arrival pattern. The reliability information of the simulation includes: the number of cases per batch, the number of batches, the number of sub-runs, the length of a sub-run, the number of start-runs. An arrival pattern is given by a probability distribution; often this is the negative exponential distribution.
- 4) The choice construction is the construction in which a choice is based on the value of the data element. The information that is added specifies the conditions to be used.

If the process definition extended with simulation data is correct then after the transformation by *Prospect* the result can directly be simulated in *ExSpect*.

3.2 Interface between COSA[®] and ExSpect

A similar simulation model as obtained with PROTONS, can be obtained from the workflow management system COSA[®] [15]. Also a number of simulation settings has to be added. Since COSA[®] is a Petri net based tool, the translation is straightforward. The application that translates a COSA[®] model to an *ExSpect* model is called CO2EX.

4 Generate Message Sequence Charts During Simulation

During simulation *ExSpect* can generate Message Sequence Charts (MSCs) [5]. We illustrate this feature by means of the Transit process, which has been modeled in *ExSpect* in a design project for European Commission. Transit is a European customs process. It controls the communication between organizations that exchange messages about a transit declaration of goods between member states of the European Union. With the model, particular interaction scenarios were generated. Fig. 5 shows a number of *ExSpect* systems. ‘NCTS’ represent roles of organizations. Fig. 5 also shows an MSC that corresponds with these roles. The MSCs are particularly useful to evaluate interaction scenarios between roles. MSCs are derived from an *ExSpect* system by monitoring the communication channels between roles. In case of the example an internal channel in the system ‘network’ is monitored.

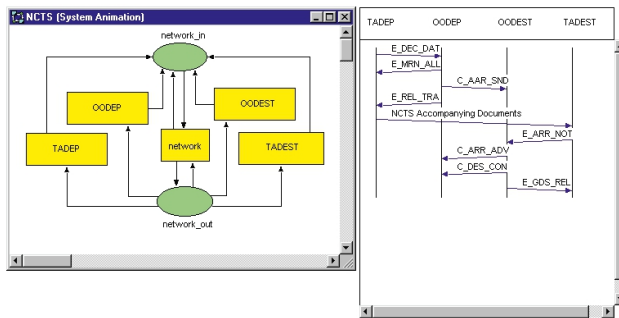


Fig. 5. Actor system and Message Sequence Chart

Each message token in a message channel is monitored: the sender and recipient of the token are registered. This results in an arrow in the MSC at the right-hand side of Fig. 5. In the *ExSpect* help file [6] a technical description of the DBO MSC is given.

UML (Unified Modeling Language), see [3] has been accepted as standard modeling language, where designers often have to conform to. UML Sequence Diagrams (SDs) model the interaction between objects. A designer can interactively construct SDs with tools. However, (s)he cannot verify these SDs with other models

like state charts or activity diagrams. Because an MSC is similar to a SD and because an MSC represents one particular trace of the dynamic behavior of a system, *ExSpect* can be used to validate these diagrams with the integrated system description in *ExSpect*.

5 *ExSpect* as a COM-Component

Recently, the *ExSpect* engine has been isolated from the other parts of *ExSpect* and made a COM component [12] with a well-defined interface. This offers the opportunity to develop applications with “*ExSpect*-inside”. In applications of this type *ExSpect* handles the control flow while the user interface and the integration with data can be handled by the window application (called the *client*). Microsoft COM components can be integrated in a number of programming environments. Some of the well known are Visual Basic, Java and C++.

A client can interact with the *ExSpect* COM server through methods and events of the server’s Application Programming Interface (API). The API intervenes in the process of the engine. A client calls methods and the server triggers events as illustrated in Fig. 6.

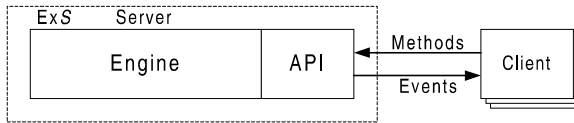


Fig. 6. Interaction between the *ExSpect* server and clients (windows applications)

The process of invoking methods and receiving events is asynchronous. After a client invokes a method it expects an event as a response. A period of time passes before the client is reached. Meanwhile the client is blocked to avoid synchronization problems.

The methods of the API are: *AddBreakpoint*, *ConsumeToken*, *Continue*, *GetTime*, *GetTokenValue*, *HaltAfterSteps*, *HaltOnTime*, *Idle*, *Init*, *LoadState*, *Pause*, *ProduceToken*, *RemoveBreakpoint*, *SaveState*, *TracePlace* and *UntracePlace*.

The events of the API are: *Consume*, *Continue*, *FireEnd*, *FireStart*, *Initialised*, *Paused*, *PlaceType*, *Present* and *Produce*.

With these methods and events and a thorough understanding of the protocol between the *ExSpect* Server and its Clients, it is possible to obtain the same monitoring and control options as one would have with the *ExSpect* Simulator and Dashboard. Presently, a tutorial on the *ExSpect* Server is developed that describes all methods, events and protocols in detail.

Fig. 7 shows an MSC, which is an example of an *ExSpect* Client - Server protocol. This protocol illustrates the following. To start client-server interaction a client calls the method *Init*. As a result a client receives an event *PlaceType* for every place in the system for which the “show in simulation” option is selected [6] and it also receives an event *Initialised* from the COM server. The event *PlaceType* provides information concerning the type of a place while the event *Initialised* indicates if the initialization has succeeded or failed. After initialization a client can start with the simulation by

calling method *Continue*. A client will receive event *Continued* as an acknowledgement. In case of more than one client, the COM server will broadcast event *Continued* to all clients. To interrupt the simulation a client has to call method *Pause*. Event *Paused* will be sent by the COM server. In this case there will also be a broadcast to all clients if there is more than one client connected to the COM server.

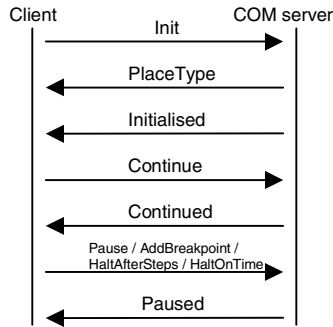


Fig. 7. ExSpect Client - Server protocol

6 Conclusions and Future Developments

This paper demonstrates a number of new features of ExSpect. In the past ten years the high level Petri-net concept appeared to be successful in solving a number of modeling problems in workflow, logistics and electronic commerce. Users requested changes of other aspects, such as user-friendliness, better integration with other tools and the capability to create prototypes rather than specifications of systems.

ExSpect addressed these requests by:

- improving the user interface and creating interfaces to other tools,
- offering the possibility to view aspects of a system, such as MSCs,
- “componentizing” the ExSpect engine and thereby offering the possibility to create user dedicated interfaces for ExSpect.

In the future the tool will be extended with other architectural views, such as a component view [11]. Also an attempt will be made to add workflow verification options to ExSpect [16].

References

1. van der Aalst, W.M.P.. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21-66, (1998).
2. van der Aalst, W.M.P.. Process-oriented Architectures for Electronic Commerce and Interorganizational Workflow. *Information Systems*, 24(8):639-671, (2000).

3. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison Wesley (1998).
4. de Crom, P.J.N.: The design of a new graphical user interface for *ExSpect*. Final report of the postgraduate program Software Technology. ISBN 90-5282-566-1. Stan Ackermans Instituut, Eindhoven (1995).
5. CCITT. CCITT Recommendation Z.120: Message Sequence Chart (MSC92). Technical report, CCITT, Geneva (1992).
6. Deloitte & Touche Bakkenist. *ExSpect*. Product Management *ExSpect* P.O. Box 23103, 1100 DP Amsterdam, The Netherlands. <http://www.exspect.com/> (1999).
7. Deloitte & Touche Bakkenist. *ExSpect* 6 User Manual. Product Management *ExSpect* P.O. Box 23103, 1100 DP Amsterdam, The Netherlands (1997).
8. Deloitte & Touche Bakkenist. *ExSpect* Language Tutorial (rel 6.0). Product Management *ExSpect* P.O. Box 23103, 1100 DP Amsterdam, The Netherlands (1997).
9. van Hee, K.M. Information Systems Engineering: a Formal Approach. Cambridge University Press (1994).
10. van Hee, K.M., Somers, L.J., Voorhoeve, M. Executable Specifications for Distributed Information Systems, in E.D. Falkenberg, P. Lindgreen (eds.), Information System Concepts: an in-depth analysis, North Holland, pages 139 - 156 (1989).
11. van Hee, K.M., van der Toorn, R.A., van der Woude, J., Verkoulen, P.: A Framework for Component Based Software Architectures. In W.M.P. van der Aalst, J. Desel, and R. Kaschek, editors, *Software Architecture Business Process Management (SABPM'99)*, pages 1-20, Heidelberg, Germany, June 1999, Forschungsbericht Nr. 390, University of Karlsruhe, Institut AIFB, Karlsruhe, Germany (1999).
12. Microsoft. Component Object Model (COM). <http://www.microsoft.com/com/>.
13. Pallas Athena. PROTON User Manual. Pallas Athena BV, Plasmolen, The Netherlands (1997).
14. Reijers, H.A. and Van der Aalst, W.M.P.: Short-term Simulation: Bridging the gap between operational control and decision making. In Proceedings of the IASTED International Conference on Modeling and Simulation 1999, May 5-8, Philadelphia. ACTA Press, Pittsburg PA, pages 417 - 421 (1999).
15. Software-Ley. COSA[®] User Manual. Software-Ley GmbH, Pullheim, Germany (1996).
16. SMIS group, Department of Computing Science, Eindhoven University of Technology. Woflan. <http://www.win.tue.nl/~woflan>.
17. Department of Computing Science, University of Aarhus. Tools on the Web. <http://www.daimi.aau.dk/PetriNets/tools/>.

LoLA

A Low Level Analyser

m

..

kschmidt@informatik.hu--berlin.de

Abstract.

p p p p p p
 p p p p p p
 pp pp
 p p (p p
 p p p
 p p
 pp

1 General Remarks

m m m m -
 m m m
 g
 - g
 g g y g
 g y m g
 m
 g m - g
 - m m g -
 y g m m -
 - m m m
 mm y m
 y m
 m y -
 m

— m m m m —
 y g m y y m y y

Example. 4 g m
 ymm 4 m m
 1 m - z
 m m g m m m y
 y m m g m m
 g g m m ym-
 m y g m g m m m y 4 y
 y m m -
 m y g g m
 m m g m m
 g g g y m m
 m m g y m
 g m mm g -
 y m

g y m y m g -
 y m y g - z
 m m m m -
 g g y g g m g y
 m y m ymm g -
 m g m

m
 y m y -
 m
 g m m g
 y
 m
 m
 m m z g m
 g
 00 y m
 0000
 m

2 How to Use LoLA

g y g m -
 m g y y g - gy m -
 - gy g m g
 g
 g g g m m g -
 g m m m ymm g m
 y g m y m
 m mmm m g mm y m
 ymm m m g g g m
 m ymm g mm g g g y g
 m y m y
 y g m g mm -
 m g g g
 CHOMSKY y 3 g mm m m g -
 g y m g m
 y m g y m y m y g
 y g g m g m g
 g m g m
 y m g m y
 g m y

```

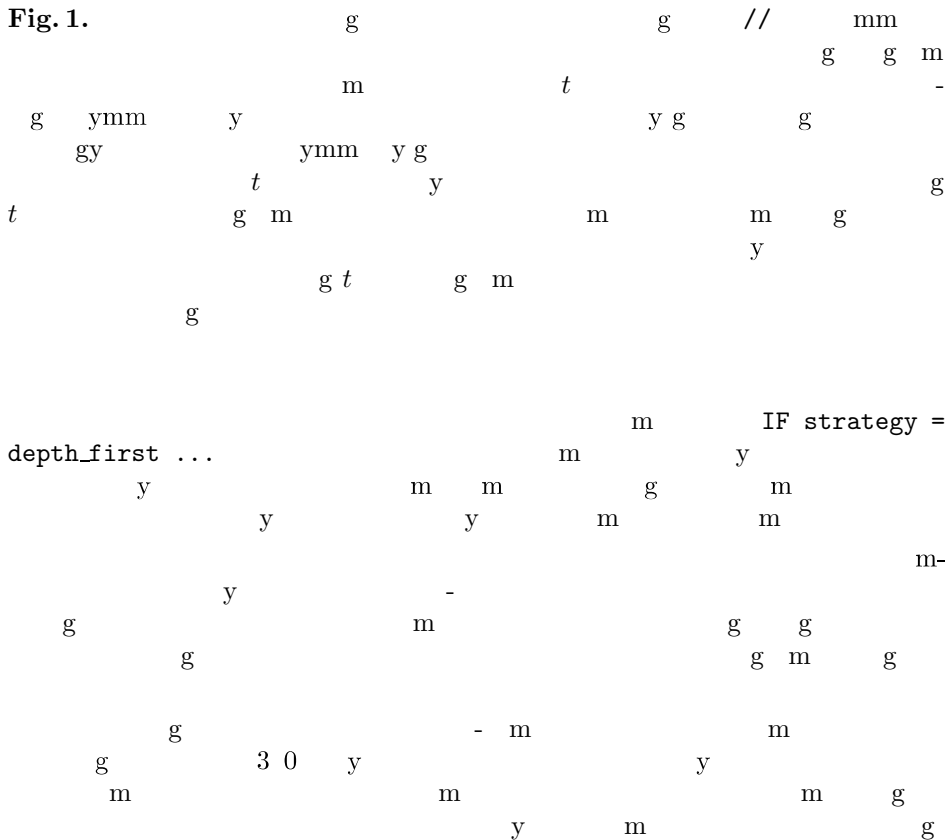
#define COVER
#define STUBBORN
#define SYMMETRY

#define GRAPH depth_first
//define GRAPH breadth_first

//define REACHABILITY
//define MODELCHECKING
//define BOUNDEDPLACE
//define BOUNDEDNET
#define DEADTRANSITION
//define REVERSIBILITY
//define HOME
//define FINDPATH
//define FULL

```

Fig. 1.



```

PLACE eating0, eating1, ( ..... )
      thinking2, thinking3, thinking4;

MARKING thinking0 : 1, thinking1 : 1, ( ..... ), fork4 : 1;

TRANSITION releaseleft0
CONSUME eating0: 1;
PRODUCE hasright0: 1, fork0: 1;

( ..... )

TRANSITION takeright4
CONSUME hasleft4: 1, fork0: 1;
PRODUCE eating4: 1;

```

Fig. 2.

g

3 State Space Exploration Techniques in LoLA

m ymm m -
y g m ymm y -
y y g y g g ymm
g y g g ymm y g m m -
y g g ymm y g m
m g m g ymm y
y ymm 0
y y g -
y m g
m y ymm y
g m y g
gy y g -
g y y
y
m g m m m
y m g y
z y m y m
ymm y
m y

m
 g m
 m
 y g
 g m
 m y y
 g
 y m
 m 0 m
 000 m
 000
 g m
 m m m m y y m -
 m
 y y
 g mm g

5 Implementation Details

m
 m g -
 y
 -
 y g m g y m
 fly g m
 g y g
 g g

5.1 State Space Management

g y
 m m m
 g
 y
 m y
 ymm m
 m ymm y g m
 m g m g
 m m g
 ymm m g
 m y
 3 m
 m m g 3
 g m y
 m -
 g y m y
 g y y m y

5.2 Firing

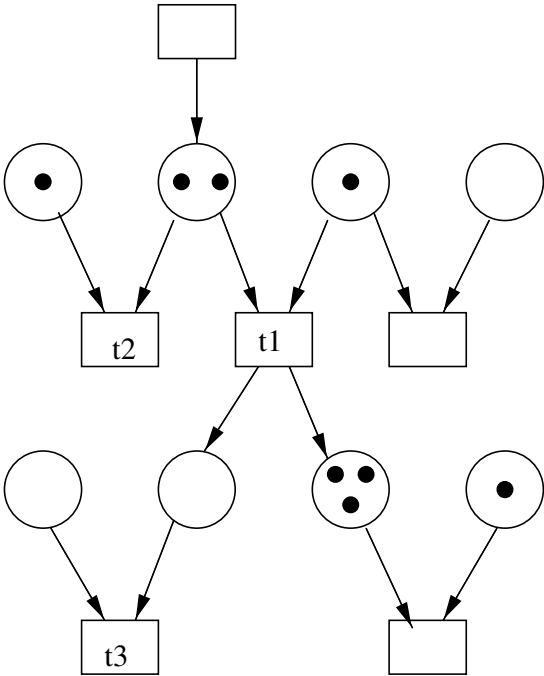
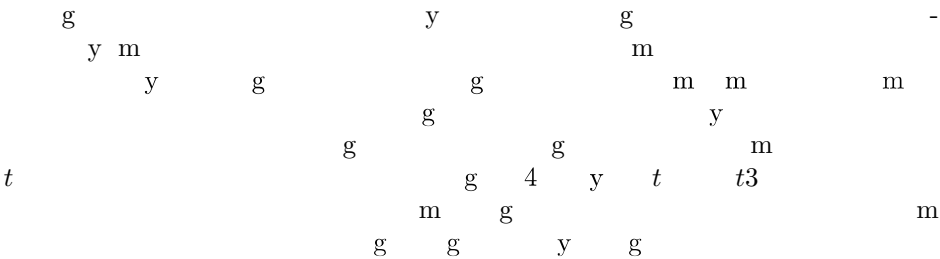
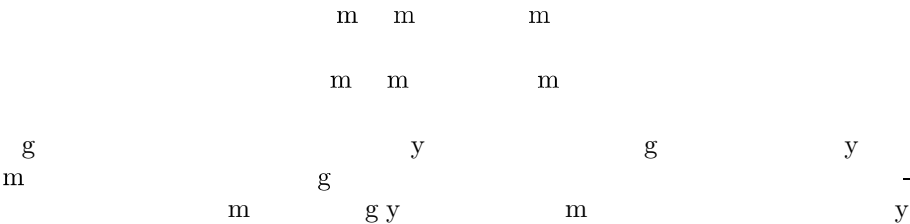


Fig. 4.

5.3 Reduction Techniques



g m g m m g
y y y g m y
g y g m m-
g m m g
ymm fl m g y
ymm y ymm y m m y
ymm ymm y -
g
y g g
m g y
y g KARP-MILLER y 3 FINKEL
y m y
m g y y
g y y g
g y

6 Performance

400 z m y g g
000 000 0
000 m 400 g m
00000 y
z ymm y y
3 m g ymm y g
g y g
ymm y g
m g m m
g g

7 Availability

g
<http://www.informatik.hu-berlin.de/~kschmidt/lola.html>

m y m
- - g -
m m m g bison
flex m g

8 Future Work

g g m - y m m
m
g g

References

p Proc. 11th International
Conference on Application and Theory of Petri nets p
p p pp
3rd Israel Symp. on the Theory of Computing
and Systems, IEEE 1995 p
p Journ. Computer and
System Sciences 4 p
Informatik-Bericht
..
Acta Informatica p
Proc. 6th
International Conference Tools and Algorithms for the Construction and Analysis
of Systems p
pp FUNDAMENTA INFORMATICA
p p 20th International Conference
on Application and Theory of Petri nets, p
p Proc. 9th
European Workshop on Application and Theory of Petri Nets,
p Petri net Newsletter 46 p
Proc.
CONCUR, LNCS 715 p

Woflan 2.0

A Petri-Net-Based Workflow Diagnosis Tool

Eric Verbeek and Wil M.P. van der Aalst

Department of Technology Management, Eindhoven University of Technology,
the Netherlands

{H.M.W.Verbeek, W.M.P.v.d.Aalst}@tm.tue.nl

Abstract. Workflow management technology promises a flexible solution facilitating the easy creation of new business processes and modification of existing ones. Unfortunately, most of today's workflow products allow for erroneous processes to be put in production: these products lack proper verification mechanisms in their process-definition tools for the created or modified processes. This paper presents the workflow diagnosis tool Woflan, which fills this gap. Using Petri-net based techniques, Woflan diagnoses process definitions before they are put into production. These process definitions can be imported from commercial workflow products. Furthermore, Woflan guides the modeler of a workflow process definition towards finding and correcting possible errors.

1 Introduction

Today's workflow management systems are ill suited to dealing with frequent changes: there are hardly any checks to assure some minimal level of correctness on the process [Aal98, AH00]. Even a simple change like adding a task can cause serious problems like deadlock or livelock. As a result, an erroneous process definition may be taken into production as a workflow, causing dramatic problems for the organization. Therefore, it is important to verify the correctness of a process definition before it becomes operational. The role of verification becomes even more important as many enterprises are making Total Quality Management (TQM) one of their focal points. For example, an ISO 9000 certification and compliance forces companies to document business processes and to meet self-imposed quality goals [IC96]. Clearly, verification of these process definitions can be used to ensure certain levels of quality.

The tool Woflan was built in response to the need for a workflow verification tool. Right from the start, three requirements have been imposed on the tool:

1. Woflan should be independent of the process definition tool used by the modeler.
2. Woflan should be able to handle complex process definitions (up to hundreds of tasks).
3. Woflan should give the modeler to-the-point diagnostic information to find and repair errors.

Based on these requirements, we decided to use Petri nets because they are a universal modeling language with a solid mathematical foundation, are close to diagramming

techniques used in practice, and have efficient analysis techniques that are already available. The primary goal of the Woflan tool is to verify a process definition, i.e., to check whether a process definition is a workflow process definition that satisfies the so-called soundness property [Aal97, Aal98].

Workflow process definition A process definition is called a *workflow* process definition if it has a single start condition (indicating the arrival of a new case), a single end condition (indicating the completion of a case) and if all tasks contribute to completing a newly-arrived case.

Soundness property A workflow process definition is called sound if it is always possible to complete a case (i.e., if it is always possible to reach the end condition), if completion is always proper (i.e., if no references to the case are left behind when it reaches the end condition), and if every task can be executed in some way.

In case the process definition is not a workflow process definition that satisfies the soundness property, Woflan's diagnostic information guides the developer towards finding and correcting the errors.

First, we explain the terminology used in this paper. Second, we describe the architecture of the tool. Third, we discuss the properties used by the tool to decide whether it is a sound workflow process definition. Fourth, we introduce the diagnosis process that helps the developer in finding and correcting the errors. Fifth, we discuss a number of import- and export filters from third party (WFMS, BPR) tools that increase Woflan's usefulness, using a diagnosis process definition as example. Sixth, we diagnose and correct the flawed diagnosis process definition. Last, we conclude with conclusions and future work.

2 Terminology

The terminology used in this paper is based on the terminology used by the WfMC [WFM96]. However, to avoid confusion within the Petri-net community, we use the term *condition* instead of *transition* to describe places. Table 1 shows the mapping from the workflow terms [WFM96] used in this paper to Petri-net related terms. For some non-standard terms a brief explanation is given.

Short-Circuited Net In [Aal97] it has been shown that a workflow net is sound if and only if that net extended with an extra transition (called *EXTENSION* in Woflan) from the sink place(s) to the source place(s) is bounded and live. This extended net is called the short-circuited net.

In the remainder of this paper, we want to avoid mentioning the short-circuited net over and over again. For this reason, some properties of a process definition are defined (see Table 1) on the short-circuited P/T net, while others are defined on the original P/T net.

Restricted Coverability Graph A restricted coverability graph (RCG) is a coverability graph (CG) except for the fact that infinite states are not expanded during construction of the RCG. Like a CG, an RCG is not uniquely defined if the net is unbounded. If no infinite states exist, an RCG equals the occurrence graph (OG) [VBA99].

Workflow	Petri net
Process definition	P/T net
Workflow process definition	Workflow (WF) net [Aal97, Aal98]
Condition	Place
Task	Transition
Start condition	Source place
End condition	Sink place
Useless task or condition	Strongly unconnected nodes in the short-circuited net
Thread of control	S-component in short-circuited net projected to places
Uniform invariant	P-invariant in short-circuited net containing only weights 0 and 1
Weighted invariant	P-invariant in the short-circuited net containing only semi-positive weights
Proper condition	Bounded place in minimal coverability graph (MCG, [Fin93]) of short-circuited net
Improper scenario	Unbounded sequence in restricted coverability graph (RCG, [VBA99]) of short-circuited net
Live task	Live transition in RCG of short-circuited net
Dead task	Dead transition in MCG of short-circuited net
Deadlock scenario	Non-live sequence in RCG of short-circuited net
Confusion	Non-free-choice cluster [DE95] in short-circuited net
AND-OR mismatch	TP-handle [EN94] in short-circuited net
OR-AND mismatch	PT-handle [EN94] in short-circuited net

Table 1. Mapping from workflow terms to Petri-net terms

Sequence A sequence is a firing sequence of minimal length (e.g., paths in the (R)CG) such that states with a given property become unavoidable (fairness etc. assumed). It is minimal in the sense that up to the last-but-one transition in the sequence (e.g., the last-but-one state in the path) the property is avoidable: the last transition in the sequence (e.g., the last edge in the RCG) makes the property unavoidable.

3 Architecture

The core of Woflan consists of Petri-net-based analysis routines. Using these routines, Woflan can verify the soundness of a given process definition. This soundness property is the minimal requirement any workflow process definition should satisfy. Because soundness is equivalent to the boundedness and liveness of the short-circuited WF net [Aal97], it can be verified using standard Petri-net techniques. Although it is possible to verify the soundness property for many process definitions in polynomial time, Woflan uses the general approach by constructing a minimal and/or restricted coverability graph. The diagnosis of the process definition is also partly based on these constructed CG's. The Woflan tool contains a number of modules:

1. One GUI module (`wofapp`),
2. One analysis module (`wofdll`) for loading, verifying and diagnosing process definitions, and

3. Three conversion modules (`scr2tpn`, `wil2tpn`, and `gwd2tpn`) for process definitions from commercial products (Cosa [SL98], Meteor [SKM], resp. Staffware [Sta97]).

The GUI- and conversion modules are implemented in the main executable (called `wofapp.exe`). To support the use of Woflan as a back-end tool, the analysis module is implemented in a separate DLL (`wofd11.dll`).

4 Properties

Soundness of a workflow process definition is equivalent to that definition being proper and live, i.e., all conditions must be proper and all tasks must be live. Therefore, to decide soundness, Woflan computes whether all conditions are proper and all tasks are live. Preceding these two properties, Woflan has to decide whether or not the process definition is indeed a workflow process definition.

4.1 Workflow

The definition of a workflow process definition is straightforward: it should be a process definition with exactly one start condition, exactly one end condition, and no useless tasks or conditions. Because these properties are of a structural nature and do not require the construction of an MCG, RCG or OG, they are relative easy to check.

4.2 Properness

Properness of conditions can be decided using the conventional method, i.e., by generating the net's MCG etc. However, because of its complexity, we would like to avoid this if possible. Fortunately, there are alternatives that are less expensive from a computational point of view: all conditions covered by threads of control, uniform invariants or weighted invariants are proper. Because a thread of control is also a uniform invariant and a uniform invariant is also a weighted invariant, we have ordered these alternatives from more desirable to less desirable.

Threads of Control From the workflow point of view, threads of control are very desirable. A workflow case typically consists of a number of documents. Each document has its own route through the workflow. A thread of control coincides with such a document route. So, if threads of control cover a workflow process definition, then each workflow case can be split into a number of documents such that each condition can be linked to some (possibly all) of these documents. If there is not such a cover, the uncovered conditions cannot be linked to any document. As a result, for a workflow process definition to be sound, both confusions and mismatches have to be present [VBA99]. Apparently, these constructions are vital to 'cure' the net from these uncovered conditions. This soundness-related property is called *interim soundness*: a process definition containing uncovered conditions is called interim sound if and only if it contains confusions and mismatches.

Diagnostic Properties If a definition contains improper conditions, Woflan computes some additional properties that can help finding and correcting the properness problem: AND-OR mismatches (they endanger properness), confusions, and improper scenarios.

4.3 Liveness

Suppose we have a process definition that can be covered by invariants (i.e., that contains no improper conditions), that can not be covered by threads of control, and that has either no confusions or no mismatches. For such a definition we can conclude that it is unsound, i.e., it contains non-live tasks.

Likewise, suppose we have a process definition containing no improper conditions and for which we have detected substates during the construction of the MCG. A reachable markings M_1 is a substate of another reachable marking M_2 iff $M_1 < M_2$. At this point, we can conclude that from M_1 the extra task EXTENSION is dead.

Otherwise, Woflan has no method yet to decide liveness without generating the OG. Note that generating this OG is only possible if the process definition contains no improper conditions.

Diagnostic Properties If a definition contains non-live tasks, Woflan computes some additional properties that can help finding and correcting the liveness problem: OR-AND mismatches (they endanger liveness), confusions, dead tasks, and deadlock scenarios.

5 Diagnosis

Based on practical experiences with earlier versions of Woflan we have developed a method for detecting errors in a workflow process. This method is supported by Woflan 2.0 and uses the diagnosis process shown in Figure 1. The diagnosis process can either be executed in-succession or step-by-step. In the latter case, dialogs are used to communicate with the user. First, we give the diagnosis process. Second, we explain one of the dialogs in detail, using the diagnosis process as shown before as example. Last, we explain the general view in detail, using again the diagnosis process itself. Please note that Figure 1 is used both as a meta-model describing the functionality of Woflan *and* as a concrete example of a workflow process.

5.1 Process

Because of the fact that we need the OG (and therefore a workflow process definition containing no improper conditions) for deciding liveness, it is obvious to decide it only if the workflow process definition has been proven to contain only proper conditions. Because the workflow properties are easy to check, Woflan starts with them. As a result, the main steps in the diagnostic process are:

1. Decide workflow,
2. Decide properness, and
3. Decide liveness.

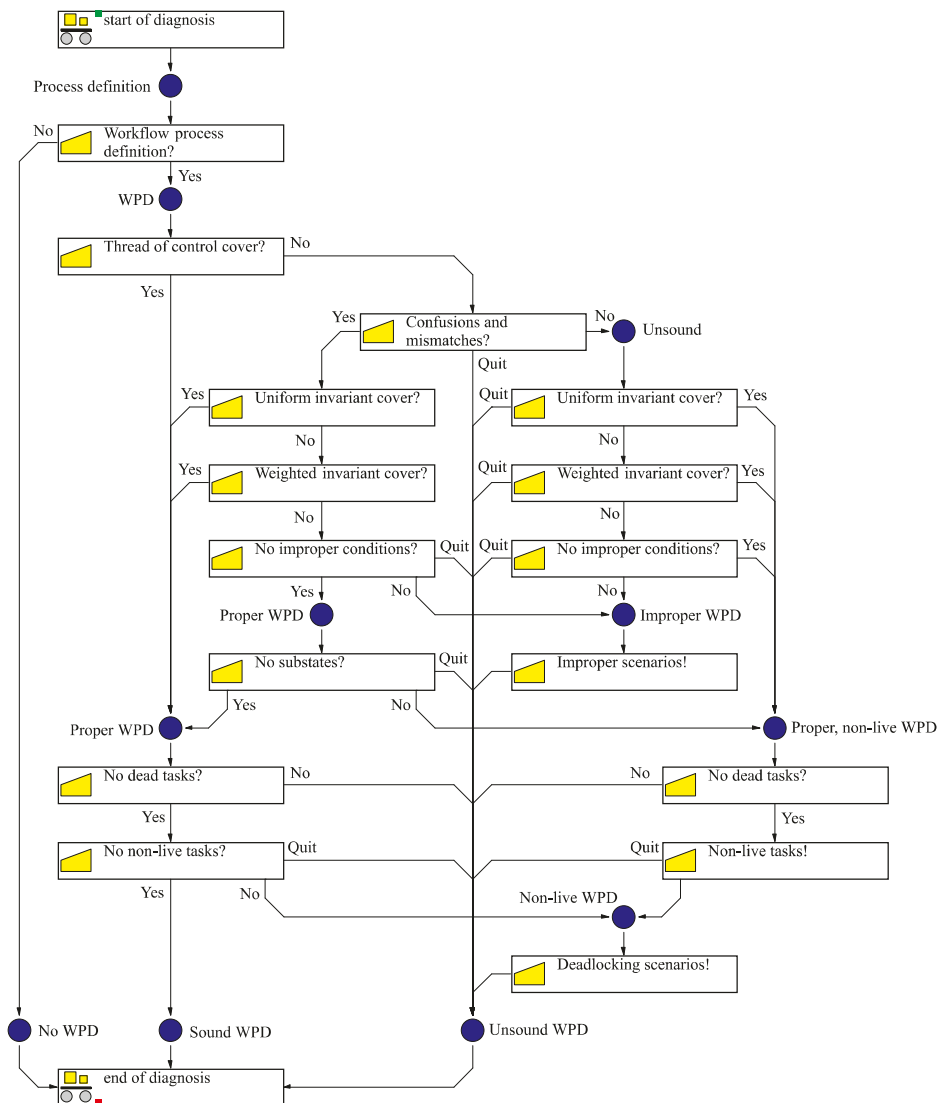


Fig. 1. Diagnosis process, modeled using Protos [Pal97]

If some step fails, there is not much use in continuing with the next step. The modeler of the process definition first has to correct the errors present.

Figure 1 shows a graphical representation of the diagnosis process definition. Note that in some cases the diagnosis process may be continued when unsoundness of the process definition has been detected. The reason for this is to collect more diagnostic information.

5.2 Dialogs

The diagnosis process uses a series of dialogs to guide the user step-by-step through the process. Depending on the diagnostic results, either a next dialog is presented or the process is finished (end of diagnosis). Properties that are likely to be of interest to the modeler are automatically unfolded. As running example, we take the diagnosis process definition as shown in Figure 1 and show the dialog concerning the thread of control cover.

Thread of Control Cover? The dialog as shown in Figure 2 shows that, using previous dialogs, Woflan has concluded that the diagnosis process definition is a workflow process definition, but that no threads of control exist. As a result, the 20 conditions of the diagnosis process definition are listed.

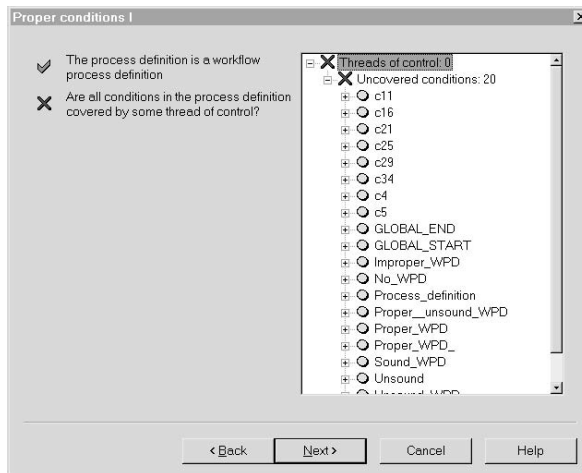


Fig. 2. Example "Thread of control cover?" dialog

5.3 Diagnosis View

The diagnosis view shows all properties of the process definition in a tree-like manner. At the root, the name of the process definition file is shown. This root node has two child nodes: the upper for the diagnosis results, the lower for the diagnostic properties. The diagnosis results node shows in brief the results on the main properties (workflow, safeness, liveness, soundness). The diagnostic properties node combines the diagnostic information from all dialogs.

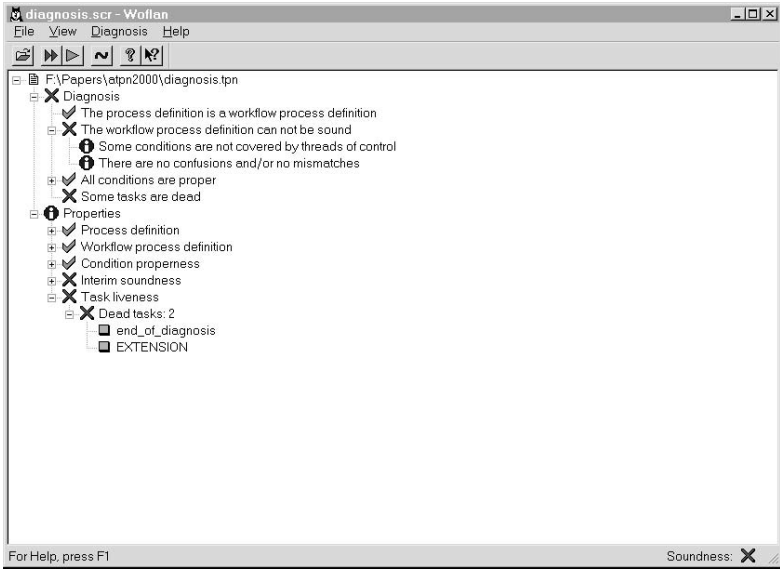


Fig. 3. Example diagnosis view

6 Links to Third-Party Software

Woflan embeds three filters to import third party process definition files:

1. For Cosa [SL98] script files (*.scr),
2. For Meteor [SKM] workflow files (*.wil), and
3. For Staffware [Sta97] (*.xfr) files.

Furthermore, the BPR tool Protos [Pal97] comes with an additional Woflan export filter, which uses Cosa script files as an intermediate format. Each embedded filter shows the results (which could be error messages) of the import process in a dialog. For readability's sake the comments are colored gray, the keywords green, and error messages red.

The dialog as shown in Figure 4 results from the diagnosis process definition (see Figure 1) that was designed using Protos, exported to Woflan (i.e., to a Cosa script file) and imported by Woflan's Cosa import filter. Note that the Cosa import filter automatically added a start condition (GLOBAL_START), an end condition (GLOBAL_END), and a token in the start condition representing a newly-arrived case.

7 Example

Apparently, the diagnosis process definition from Figure 1 is unsound. In this case, particularly the dead tasks are of interest. Note that the short-circuiting task EXTENSION is dead. As a result, all tasks are non-live. The task EXTENSION can only be dead if task end_of_diagnosis is dead, which is dead because it acts as an AND-join instead

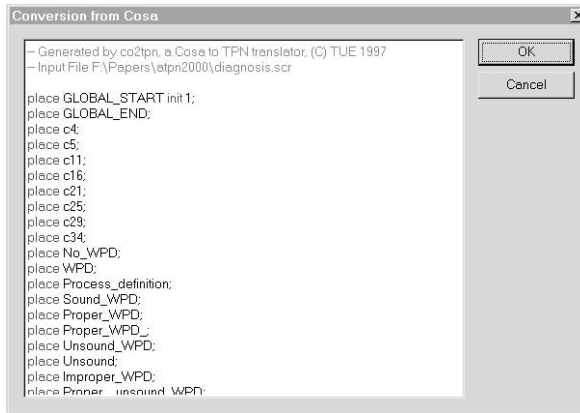


Fig. 4. Example import filter dialog

of an OR-join. Protos' export filter and Woflan's import filter allow to have this property changed in Protos, both filters can handle a task which acts as an OR-join (split) instead of as an AND-join (split). After changing this property in Protos, exporting it to Woflan and importing the resulting Cosa script file, the diagnosis process appears to be a sound workflow process definition. Although this error seems trivial, taking a workflow with such a flawed process definition into production will result in much irritation and agony: prevention is better than cure. Also note that real world examples are not as straightforward as the workflow process definition shown in Figure 1.

8 Conclusions and Future Work

For several commercial WFMS/BPR products, Woflan can be used to verify a process definition, checking both syntactic (cf. Section 4.1) and behavioral (cf. Sections 4.2 and 4.3) properties. By using Woflan and its state-of-the-art techniques it is possible to prevent that an unsound workflow is taken into production.

In the nearby future we hope to extend Woflan with two more features: transition invariants and visualization.

A *sound* workflow process definition is covered by non-negative transition invariants. If the process definition is safe, a task that is not covered by these invariants cannot be live. In a next version of Woflan we hope to use this property to avoid the use of the OG, if possible.

We also would like to visualize the diagnosis results in some intuitive way, using Petri nets of course. If possible, we even want to visualize these results in the WFMS/BPR tool the modeler is using. To support this use of Woflan as a back-end tool, we separated the analysis techniques from the rest of the tool.

Woflan can be downloaded from [VA].

9 Acknowledgements

The authors wish to thank Twan Basten and Hajo Reijers for their fruitful remarks and suggestions.

References

- [Aal97] W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets 1997, Proceedings*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426, Toulouse, France, June 1997. Springer, Berlin, Germany, 1997.
- [Aal98] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
- [AH00] W.M.P. van der Aalst and A.H.M. ter Hofstede. Verification of Workflow Task Structures: A Petri-net-based Approach. *Information Systems*, To appear, 2000.
- [DE95] J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.
- [EN94] J. Esparza and M. Nielsen. Decidability Issues for Petri Nets - a Survey. *Journal of Information Processing and Cybernetics*, 30(3):210–242, 1994.
- [Fin93] A. Finkel. The Minimal Coverability Graph for Petri Nets. In G. Rozenberg, editor, *Advances in Petri Nets 1993*, volume 674 of *Lecture Notes in Computer Science*, pages 210–243. Springer, Berlin, Germany, 1993.
- [IC96] R.R.A. Issa and R.F. Cox. Using Process Modeling and Workflow Integration to gain (ISO 9000) Certification in Construction. In *CIB W89 Beijing International Conference on Construction, Modernization, and Education*, Beijing, China, 1996.
- [Pal97] Pallas Athena. *Protos User Manual*. Pallas Athena BV, Plasmolen, The Netherlands, 1997.
- [SKM] A. Sheth, K. Kochut, and J. Miller. Large Scale Distributed Information Systems (LSDIS) laboratory, METEOR project page. <http://lstdis.cs.uga.edu/proj/meteor/meteor.html>.
- [SL98] Software-Ley. *COSA User Manual*. Software-Ley GmbH, Pullheim, Germany, 1998.
- [Sta97] Staffware. *Staffware 97 / GWD User Manual*. Staffware Plc, Berkshire, UK, 1997.
- [VA] H.M.W. Verbeek and W.M.P. van der Aalst. Woflan Home Page. <http://www.win.tue.nl/~woflan>.
- [VBA99] H.M.W. Verbeek, T. Basten, and W.M.P. van der Aalst. Diagnosing Workflow Processes using Woflan. Computing Science Report 99/02, Eindhoven University of Technology, Eindhoven, The Netherlands, 1999.
- [WFM96] WPMC. Workflow Management Coalition Terminology and Glossary (WFMC-TC-1011). Technical report, Workflow Management Coalition, Brussels, 1996.

Author Index

7 7

7

7

..

7

..

7

7

7

7

7

7

.. ..

7